UNIVERSITÉ DE SHERBROOKE
Faculté de génie
Département de génie électrique et de génie informatique

Transport coopératif d'un objet par deux robots humanoïdes dans un environnement encombré

Mémoire de maîtrise
Spécialité : génie électrique

Antoine Rioux

Jury : Wael Suleiman (directeur de recherche)
François Michaud (rapporteur)
Jean-Bernard Hayet (examinateur externe)

Sherbrooke (Québec) Canada Novembre 2016

RÉSUMÉ

Il y a présentement de la demande dans plusieurs milieux cherchant à utiliser des robots afin d'accomplir des tâches complexes, par exemple l'industrie de la construction désire des travailleurs pouvant travailler 24/7 ou encore effectuer des operation de sauvetage dans des zones compromises et dangereuses pour l'humain. Dans ces situations, il devient très important de pouvoir transporter des charges dans des environnements encombrés. Bien que ces dernières années il y a eu quelques études destinées à la navigation de robots dans ce type d'environnements, seulement quelques-unes d'entre elles ont abordé le problème de robots pouvant naviguer en déplaçant un objet volumineux ou lourd. Ceci est particulièrement utile pour transporter des charges ayant de poids et de formes variables, sans avoir à modifier physiquement le robot. Un robot humanoïde est une des plateformes disponibles afin d'effectuer efficacement ce type de transport. Celui-ci a, entre autres, l'avantage d'avoir des bras et ils peuvent donc les utiliser afin de manipuler précisément les objets à transporter.

Dans ce mémoire de maîtrise, deux différentes techniques sont présentées. Dans la première partie, nous présentons un système inspiré par l'utilisation répandue de chariots de fortune par les humains. Celle-ci répond au problème d'un robot humanoïde naviguant dans un environnement encombré tout en déplaçant une charge lourde qui se trouve sur un chariot de fortune. Nous présentons un système de navigation complet, de la construction incrémentale d'une carte de l'environnement et du calcul des trajectoires sans collision à la commande pour exécuter ces trajectoires. Les principaux points présentés sont : 1) le contrôle de tout le corps permettant au robot humanoïde d'utiliser ses mains et ses bras pour contrôler les mouvements du système à chariot (par exemple, lors de virages serrés) ; 2) une approche sans capteur pour automatiquement sélectionner le jeu approprié de primitives en fonction du poids de la charge ; 3) un algorithme de planification de mouvement qui génère une trajectoire sans collisions en utilisant le jeu de primitive approprié et la carte construite de l'environnement ; 4) une technique de filtrage efficace permettant d'ignorer le chariot et le poids situés dans le champ de vue du robot tout en améliorant les performances générales des algorithmes de SLAM (Simultaneous Localization and Mapping) défini ; et 5) un processus continu et cohérent d'odométrie formés en fusionnant les informations visuelles et celles de l'odométrie du robot. Finalement, nous présentons des expériences menées sur un robot Nao, équipé d'un capteur RGB-D monté sur sa tête, poussant un chariot avec différentes masses. Nos expériences montrent que la charge utile peut être significativement augmentée sans changer physiquement le robot, et donc qu'il est possible d'augmenter la capacité du robot humanoïde dans des situations réelles.

Dans la seconde partie, nous abordons le problème de faire naviguer deux robots humanoïdes dans un environnement encombré tout en transportant un très grand objet qui ne peut tout simplement pas être déplacé par un seul robot. Dans cette partie, plusieurs algorithmes et concepts présentés dans la partie précédente sont réutilisés et modifiés afin de convenir à un système comportant deux robot humanoides. Entre autres, nous avons un

algorithme de planification de mouvement multi-robots utilisant un espace d'états à faible dimension afin de trouver une trajectoire sans obstacle en utilisant la carte construite de l'environnement, ainsi qu'un contrôle en temps réel efficace de tout le corps pour contrôler les mouvements du système robot-objet-robot en boucle fermée. Aussi, plusieurs systèmes ont été ajoutés, tels que la synchronisation utilisant le décalage relatif des robots, la projection des robots sur la base de leur position des mains ainsi que l'erreur de rétroaction visuelle calculée à partir de la caméra frontale du robot. Encore une fois, nous présentons des expériences faites sur des robots Nao équipés de capteurs RGB-D montés sur leurs têtes, se déplaçant avec un objet tout en contournant d'obstacles. Nos expériences montrent qu'un objet de taille non négligeable peut être transporté sans changer physiquement le robot.

Mots-clés : Robots humanoides, système multi-robots, SLAM, synchronization, vision, tâches coopératives

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ACRONYMES

Acronyme	**Definition**
COG	Center of Gravity
COM	Center of Mass
DARPA	Defense Advanced Research Projects Agency
DCZMP	Dynamically Complemental Zero Moment Point
MCL	Monte Carlo Localization
MDP	Markov Decision Process
PID	Proportional Integral Derivative
SLAM	Symultaneous Localization and Mapping
ZMP	Zero Moment Point

CHAPITRE 1

Introduction

Les robots humanoïdes sont de plus en plus utilisés dans divers milieux. Comme leur nom l'indique, les robots humanoïdes sont des robots à ressemblance humaine. Ils ont un corps ressemblant à celui d'un humain, généralement possédant un torse, une tête, deux bras et deux jambes. L'intérêt d'utiliser des robots humanoïdes est de pouvoir utiliser à la fois des bras pour pousser, transporter, ouvrir et faire des tâches motrices fines, et des jambes permettant de s'adapter à presque tout type de terrains.

Le premier robot de ce genre a été développé par l'université de Waseda au Japon en 1973 [Kato, 1973]. Ce robot, nommé WABOT-1, était capable d'effectuer une marche bipède. Plusieurs années plus tard, Honda développe P2 [Hirai *et al.*, 1998], le premier robot humanoïde à pouvoir marcher sur deux jambes de façon autonome. Ce robot fait partie de la série P de Honda et est un précurseur au célèbre modèle ASIMO [Sakagami *et al.*, 2002] de la même compagnie. Aujourd'hui, les robots humanoïdes sont capables de marcher sur des surfaces inégales incluant des obstacles et peuvent maintenir leur équilibre même s'ils sont déstabilisés par une force externe.

Les robots humanoïdes peuvent être plus appropriés que des robots roulants dans plusieurs situations, comme l'industrie du divertissement avec des sports robotiques, le secteur de la construction avec des travailleurs de chantiers présents 24/7 ainsi que l'accès à des lieux dangereux et missions de sauvetages évitant de mettre en danger des êtres humains. De manière plus concrète, la compétition de soccer robotique annuelle RoboCup promeut le développement de robots humanoïdes de toutes tailles. Un example d'épreuves pour cette compétition est montré à la figure 1.1. Il s'agit d'un tournoi international permettant a des dizaines d'équipes de faire jouer une équipe de robots afin de présenter leurs recherches en matière d'intelligence artificielle et contrôle humanoïde. Le but ultime de cette initiative est de voir une équipe de robots humanoïdes autonomes de taille humaine jouer au soccer et gagner contre une équipe professionnelle d'humains d'ici 2050.

Dans le contexte de sécurité et de missions de sauvetages, les robots humanoïdes peuvent également être très utiles. En effet, ayant une morphologie semblable à l'humain, ils sont beaucoup plus adaptés aux environnements destinés aux humains (escaliers, échelles, valves, portes, etc.). C'est pourquoi la DARPA (Defense Advanced Research Projects

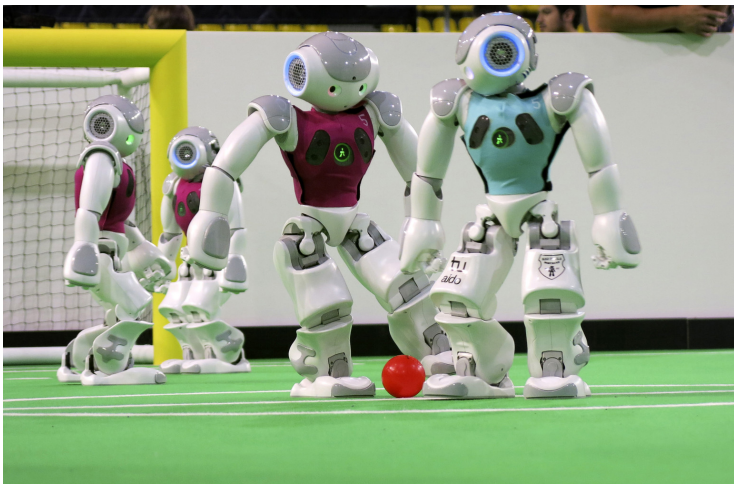


Figure 1.1 Plusieurs robots Nao participant à la compétition RoboCup.[1]

Agency) a lancé le programme DARPA ROBOTICS CHALLENGE montré à la figure 1.2. Cette compétition a vu le jour peu de temps après la catastrophe de la centrale nucléaire Fukushima. Le niveau de radiation était tellement élevé qu'aucun humain ne pouvait s'approcher de la centrale pour refroidir le réacteur. Le but de cette compétition de robotique est de promouvoir le développement de robots humanoïdes pouvant intervenir sur des sites affectés par une catastrophe. Étant donné l'environnement dans lequel le robot doit être actif, il doit être capable d'exécuter des tâches complexes dans un environnement dangereux, dégradé et spécialement construit pour des humains. Cette compétition est offerte aux laboratoires du monde entier et le DARPA fournit même un robot humanoïde complet aux équipes désirant participer et ne possédant pas l'expertise ou le financement pour fabriquer une plate-forme physique. L'objectif est le développement de robots pour remplacer les êtres humains sur des sites qui comportent un risque élevé, mais dont l'accès est trop difficile ou l'environnement est mal adapté pour des robots conventionnels.

Dans ces lieux dangereux, il peut s'avérer nécessaire pour le robot de déplacer un objet très lourd ou très grand afin qu'il puisse effectuer sa tâche. Une solution possible est d'augmenter la puissance des moteurs. Par contre, puisque l'augmentation de cette puissance est liée à une augmentation de la grosseur, du poids, du prix et de la quantité d'énergie utilisé par le moteur, il y a une limite à cette possibilité. Une autre solution est de refaire une planification de la navigation pour trouver un chemin alternatif qui ne serait pas encombré. Par contre, comme illustré à la figure 1.3, la solution alternative peut s'avérer être beaucoup plus longue. Cela résulte en une perte d'efficacité importante qui nuit au bon fonctionnement du système. Aussi, parfois il est impossible d'accéder à la zone désirée

[1] http ://whenonearth.net/wp-content/uploads/2013/09/Toby-Sterling-huffpost1.jpg

Figure 1.2 Image officielle de la compétition DARPA Robotics Challenge.[2]

dû à la présence d'obstacles. Dans le cas d'une opération de sauvetage, cela signifierait alors qu'il ne serait pas possible d'aider une personne en détresse positionnée derrière ces obstacles. Un autre approche pour résoudre ce problème est plutôt en effectuant des tâches coopératives entre plusieurs robots. C'est d'ailleurs cette dernière approche qui est explorée dans cette recherche. La question de recherche est : comment peuvent coopérer plusieurs robots humanoïdes pour déplacer des objets très lourds ou très grands dans un environnement encombré ?

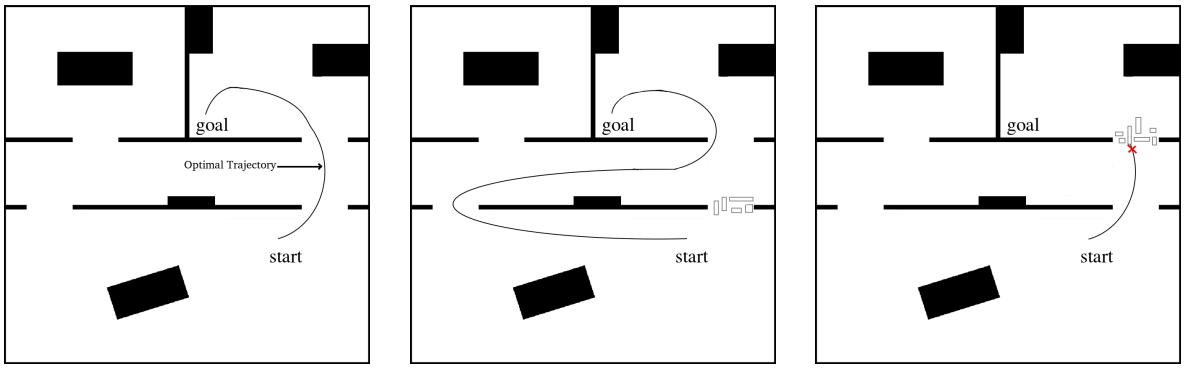


Figure 1.3 Trajectoire avec différents obstacles, de gauche à droite : Trajectoire optimale, trajectoire longue, trajectoire impossible

Les contributions apportées dans ce projet de maîtrise sont les suivantes : (1) un système de contrôle de corps complet qui permet à un robot humanoïde d'utiliser ses mains et ses bras pour contrôler les mouvements d'un système robot-objet-robot (e.g., virage serré) ; (2) un algorithme de planification de mouvement à deux robots humanoïdes pour trouver une trajectoire sans collisions en utilisant une carte construite en temps réel ; (3) une

[2]http ://www.darpa.mil/uploadedImages/Content/OurWork/TTO/ Programs/RoboticsChallenge/Graphic-DARPARoboticsChallenge.jpg

odométrie continue et fiable formée de la fusion d'une odométrie visuelle et de l'odométrie interne du robot ; (4) un système de synchronisation de mouvement utilisant la projection des robots en passant par le positionnement des mains et l'erreur de position et rotation déterminées visuellement par les robots face à face.

Ce document est séparé en plusieurs sections. Le chapitre 2 est une revue bibliographique de ce qui a déjà été fait en robotique humanoïde, en cooperation entre robots et en vision, ainsi qu'une critique des différentes techniques utilisées. Le chapitre 3 et le chapitre 4 présentent les articles qui ont été publiés durant cette maîtrise. Plus précisement, le chapitre 3 se concentre sur l'utilisation d'un seul robot pour déplacer une charge lourde, alors que le chapitre 4 ajoute l'aspect de coopération et de synchronisation entre deux robots. Ces articles décrivent l'ensemble des développements importants, résultats et contributions apportés par ces travaux de recherche. Finalement, une conclusion permet de résumer le projet et les contributions originales apportées, ainsi que de présenter la vision de ce projet pour des travaux futurs. En annexe, les articles de conférence présentés à IROS 2015 [Rioux et Suleiman, 2015] et Humanoid 2015 [Rioux *et al.*, 2015], issus des présents travaux, sont aussi fournis.

CHAPITRE 2

Tâches coopératives en robotique

De nombreuses techniques ont été étudiées, développées et testées au cours des dernières années dans le domaine de la collaboration robotique. En effet, bien que l'intérêt et l'utilisation de robots humanoïdes aient récemment grandi dans les laboratoires de recherche, ces techniques ont été appliquées tout d'abord sur des robots non humanoïdes.

2.1 Transport par robots à roues

Le transport d'objets par plusieurs robots non humanoïdes a déjà été réalisé dans le passé. Dans le système proposé par Kube et Bonabeau [Kube et Bonabeau, 2000], plusieurs robots mobiles de petite taille se positionnent autour d'un objet très large afin de le déplacer en le poussant. Le comportement de ces robots est inspiré par la façon dont les fourmis transportent des objets beaucoup plus grosses qu'elles-mêmes en utilisant leur grand nombre et en coopérant dans un but commun. Les robots se déplacent jusqu'à l'objet à bouger, puis tournent autour de façon antihoraire jusqu'à ce qu'une configuration soit adoptée pour pousser vers une cible délimitée par une lumière au sol. Lorsqu'il est observé que la direction choisie n'est pas tout à fait exacte, les robots s'arrêtent, se repositionnent et recommencent leur poussée par la suite. Il s'agit d'un des premiers exemples de coopération entre plusieurs robots qui a été réalisé.

D'autres robots mobiles à roues ont été utilisés pour déplacer un object massif. Dans [Ota *et al.*, 1995], des robots à roues transportent l'objet en soutenant la totalité de son poids au lieu de le pousser. La méthode de coopération utilisée pour naviguer parmi des obstacles consiste en une stratégie de modification de prise de l'objet combinée à un planificateur de trajectoires en deux étapes, une pour l'objet et l'autre pour les robots. Pour planifier la trajectoire de l'objet, des ressorts et des amortisseurs virtuels sont ajoutés entre l'objet et le but. Les forces et moments créés sont ensuite utilisés pour générer le plan de l'objet. Puis, étant donné qu'un seul robot tient fermement l'objet à tout moment afin d'éviter d'importants retours de force, les autres robots suivent la trajectoire et alternent entre tenir fermement et relâcher l'objet tout au long du plan afin de se repositionner autour de lui de manière plus optimale.

La même équipe de chercheurs a ensuite proposé une amélioration de leur système [Miyata *et al.*, 1997]. Une des modifications apportées au système est qu'à tout moment durant le transport, plusieurs robots maintiennent l'objet pour le déplacer pendant que les autres se repositionnent. Également, un des robots se fait assigner un rôle de superviseur et est en charge de mettre en place le plan global centralisé de l'environnement, pendant que les autres s'occupent de leur plan local. Cette approche remplace le contrôle décentralisé sur chaque robot dans [Ota *et al.*, 1995]. Un système de contrôle centralisé peut s'avérer particulièrement efficace pour faire de la coopération. En effet, au lieu que les individus échangent constamment la totalité de leurs informations avec les autres pour calculer séparément un plan, une seule unité de calcul reçoit l'ensemble des données et partage le plan global. De plus, cette structure peut être déléguée à un serveur séparé lorsque le processeur de chaque robot n'est pas suffisamment performant.

Afin d'avoir un meilleur contrôle sur l'objet transporté, un bras primitif à un degré de liberté a été ajouté à un groupe de robots par [Wang *et al.*, 1999]. Bien que ce bras soit limité, le point de contact de chaque robot à l'objet permet de calculer la position désirée du groupe de robots. Afin d'éviter les retours de forces indésirables de cette coopération, des stratégies de positionnement sont proposées. Premièrement, chaque robot peut appliquer une force dans la direction de son contact avec l'objet. Deuxièmement, chaque robot peut subir une force dans toutes les autres directions. Finalement, la direction de contact de chaque robot est déterminée pour déplacer l'objet.

Également, d'autres robots mobiles ont été utilisés pour déplacer une boîte dans un environnement 3D avec plusieurs obstacles, nécessitant de changer sa configuration dans l'espace [Yamashita *et al.*, 2003]. Dans ce travail, des bras primitifs sur chaque robot ont été ajoutés pour manipuler la boîte avec une certaine précision. Plusieurs mouvements primitifs sont vérifiés afin d'assurer la stabilité de la boîte tout en lui permettant de naviguer parmi les obstacles. Les manipulations possibles sont par contre restreintes à une translation au sol sur le plan, une rotation autour de l'axe Z, un repositionnement des robots autour de la boîte et tourner la boite sur elle-même au sol.

2.2 Cooperation humain-robot

Au lieu de fonctionner de manière autonome pour déplacer des objets, certains groupes de robots visent plutôt à aider le travail d'un humain qui désire transporter de lourdes charges. C'est le cas de [Hirata et Kosuge, 2000] qui propose l'utilisation de plusieurs robots omnidirectionnels appelés DR Helpers. L'algorithme de contrôle développé permet

aux robots d'agir à titre de roulettes pivotantes de type caster. Lorsqu'une force est appliquée dans une direction, le groupe détermine la commande à appliquer pour l'ensemble du système, permettant à l'humain de bouger l'objet sans force. Cette commande est calculée dynamiquement, selon la nature du mouvement désiré. En effet, un mouvement rapide permet au contrôle de type caster de diminuer la divergence sur une ligne droite en empêchant des changements brusques de direction, alors que pour les mouvements précis, le contrôle devient quasi omnidirectionnel.

Dans le cas où les robots ne supportent pas la totalité du poids et que l'humain coopérant avec l'aide mécanique doit aussi soutenir la tâche, il est primordial de gérer la distribution de ce poids entre tous les participants. Pour ce faire, un robot à roues possédant deux bras articulés a été utilisé [Lawitzky *et al.*, 2010]. Dans ce travail, les forces sur le plan bidimensionnel (X,Y) ainsi que le couple autour de l'axe Z (aux points de contact des mains du robot et aux points de contact des mains de l'humain) sont utilisées pour calculer l'effort nécessaire à la réalisation du plan. Lorsqu'un plan est trouvé et que l'objet est prêt à être déplacé, un coefficient λ permet de déterminer de quelle façon l'effort sera distribué entre les participants. Pour $\lambda = 0$, l'effort sera équitablement distribué et donc le robot et l'humain devront fournir les mêmes forces à l'objet. Avec λ fortement négatif, l'effort est maximal pour le robot et minimal pour l'humain uniquement lorsque le robot ne peut exécuter une section seul. Finalement, un λ fortement positif résulte en la distribution inverse de charges et l'humain fait tout le travail de déplacement, tandis que le robot ne fait que soutenir l'objet. Les résultats démontrent que l'erreur est moindre dans la direction du mouvement lorsque l'effort et la participation du robot est plus grande et n'affecte pas l'erreur dans les directions perpendiculaires au mouvement.

Bien sûr, il ne s'agit pas de la seule technique pour distribuer l'effort entre un robot et un humain. En effet, [Suda et Kosuge, 2002] se sont servis d'un robot nommé MR Helper pour proposer leur approche. Le MR Helper à base mobile omnidirectionnelle possède deux bras manipulateurs de 7 degrés de libertés, des capteurs de forces et couples à six axes à chaque main ainsi qu'une vision stéréoscopique dans la tête. Le robot peut fonctionner en deux modes distincts lors du transport d'objet lourd, soit en mode coopératif ou en mode autonome. De manière similaire au travail de [Lawitzky *et al.*, 2010], le mode coopératif suit l'humain en soutenant l'objet sans y appliquer de forces d'intentions, il se fie donc aux données de ces capteurs de forces pour suivre l'humain. Ce mode s'active automatiquement lorsque l'humain est près du robot. Le mode autonome, qui s'active lorsque l'humain s'éloigne du robot, quant à lui se base sur les informations visuelles de l'intention de l'humain pour prédire ses mouvements et soutenir adéquatement l'humain dans la tâche.

Le but visé dans ce travail est d'unifier les informations visuelles avec les capteurs de forces pour éviter les changements brusques de mode de contrôle lors du déplacement. Ces informations visuelles sont alors traduites en forces virtuelles attractives et correctives. Le mouvement du robot est donc continuellement contrôlé en force et la transition entre les modes devient imperceptible.

Bien que MR Helper possède une tête, des bras et un torse, il n'en reste pas moins qu'il se déplace avec une base roulante. Pour obtenir une collaboration similaire à une collaboration humain-humain, ressemblant davantage à une coopération humanoïde-humanoïde, le robot humanoïde Nao [Gouaillier *et al.*, 2009] a été utilisé pour coopérer avec un humain pour déplacer une table [Berger *et al.*, 2013]. Dans cette approche, uniquement les capteurs internes du Nao sont considérés. Puisque le Nao ne possède pas de capteurs de forces/torques et que les informations des caméras sont ignorées, il s'agit plutôt des capteurs de pression sous les pieds ainsi que des valeurs des angles de moteurs passifs qui servent de base pour le contrôle. Des informations supplémentaires sont aussi calculées, telles que le centre de masse, le centre de pression et le vecteur de réaction du sol. Dans la relation maître/esclave dans laquelle le Nao participe, il ne peut que se déplacer vers l'avant, vers l'arrière, sur le côté ou se lever. Après avoir créé un modèle de tous ces mouvements primitifs, un algorithme d'apprentissage avec un noyau périodique tente de déduire et de faire un liens entre les informations des capteurs, les informations calculées et les modèles de mouvements. Lorsque l'un des mouvements de base est détecté, l'amplitude de la perturbation sert à déterminer la vitesse d'exécution de celui-ci. Si elle est trop petite, le mouvement est considéré comme du bruit, sinon la vitesse augmente par pallier selon l'amplitude, gérée par un contrôleur PD.

Le HRP2 [Kaneko *et al.*, 2004] est un robot humanoïde à taille réelle ayant des capteurs de forces dans les mains et quatre caméras dans la tête. Il a été utilisé pour effectuer une tâche de collaboration avec un humain. Cette tâche consiste à repérer un objet large et le déplacer avec un humain [Yokoyama *et al.*, 2003]. La figure 2.1 illustre un exemple de cette collaboration entre un humain et un robot HRP2. L'humain et le robot sont presque de la même taille et les deux peuvent se déplacer de manière bipède omnidirectionnelle. Le HRP2 est tout d'abord principalement contrôlé par la voix. En effet, le partenaire donne des ordres au robot pour se placer en position devant le panneau à déplacer, le lever, se mettre en marche, le déposer, etc. Pendant ce temps, le HRP2 interprète les commandes vocales et répond à l'humain pour confirmer son bon fonctionnement. Au moment de la phase de positionnement vers l'objet, les caméras dans la tête du robot lui permettent d'identifier les coins du panneau à l'aide de la librairie de traitement d'images VVV [Tomita

et al., 1998] et puis se positionne pour la prise de celui-ci. Lors du mouvement proprement dit, les informations de ses capteurs de forces situés dans les mains sont utilisées avec un contrôle par impédance. Ceci ajoute un facteur d'amortissement qui permet d'absorber les perturbations indésirables et les mouvements brusques de l'humain.

Figure 2.1 Collaboration entre un humain et un robot humanoïde HRP2 à grandeur humaine.[1]

Aussi, des recherches ont été effectuées dans le cadre d'un transport d'objet de grande taille dans un environnement contraint [Bussy *et al.*, 2012]. Dans un premier temps, l'humain dirige le robot HRP2 qui est forcé de suivre les mouvements de l'humain en se servant des informations fournies par les capteurs de forces dans ses mains. Ces informations déterminent laquelle des cinq actions de base (avancer, tourner, arrêter, marcher de côté, avancer en tournant) le robot va effectuer afin de suivre l'humain. Dans un second temps, les rôles sont inversés et l'humain suit les actions du robot. Dans les deux cas, un contrôle de type maître/esclave est utilisé, c'est-à-dire que l'humain ou le robot joue le rôle du maître qui donne les commandes alors que l'autre joue le rôle de l'esclave et doit se contenter de suivre le maître le plus fidèlement possible. L'utilisation du mode de contrôle par maître/esclave comporte plusieurs défauts en ce qui concerne la coopération entre robots. En effet, un délai non négligeable est présent entre chaque mouvement qu'effectue le maître et la réponse au mouvement de l'esclave. De plus, une erreur d'interprétation du mouvement peut facilement causer une déstabilisation du système en boucle fermée. Finalement, la plupart des robots n'ont pas de capteurs de force dans les mains, car ceux-ci sont relativement gros et dispendieux.

[1] http ://www.electronichouse.com/images/slideshow/kawada-panel-web.jpg

2.3 Transport par robots humanoides

Dans un autre type d'interaction avec un humain, le HRP2 déplace plutôt l'humain assis sur une chaise roulante et suit ses indications [Nozawa *et al.*, 2008]. Dans un premier temps, le robot utilise un filtre à particules et des indices visuels pour repérer les lignes droites de la chaise roulante et plus particulièrement déterminer précisément la position des poignées. Une fois la chaise roulante en main, le HRP2 interagit avec l'humain soit par des commandes vocales, soit par des commandes visuelles. La personne tourne son visage vers le robot pour indiquer son intention de communiquer puis pointe dans une direction. La direction générale du bras indique vers où se diriger jusqu'à la réception d'une commande d'arrêt. Pour déplacer un objet aussi lourd en poussant, le HRP2 déplace son ZMP devant lui, ce qui crée une force vers l'avant causée par le poids du robot.

Dans un travail similaire, [Takubo *et al.*, 2005] se servent aussi d'un robot HRP2 afin de pousser un objet lourd avec un contrôle par impédance des bras. Cependant, l'objet poussé est une table à roulette et le HRP2 est contrôlé par une nouvelle méthode nommée "Dynamically Complemental Zero Moment Point" ou DCZMP. Le DCZMP se base sur la trajectoire du ZMP, qui est très souvent utilisée en robotique humanoïde pour gérer la stabilité, pour modifier le centre de masse (COM) de manière dynamique en se fiant aux forces externes agissant sur les mains du robot. La figure 2.2 montre un exemple de son utilisation. Leurs expériences montrent que lorsque les forces externes de réactions de l'objet sur le robot ne sont pas considérées dans le contrôle, comme ça serait le cas avec le ZMP traditionnel, le robot est déséquilibré et chute.

Un autre travail impliquant un HRP2 poussant un objet grand et lourd est proposé par [Harada *et al.*, 2007]. Cette fois ci, les phases de poussées et de marche sont séparées. Le robot pousse seulement lorsqu'il est stable et en double support. Il peut donc marcher séparément sans considérer les forces de rétroaction de l'objet. Lorsque l'objet se fait déplacer, un contrôle des bras par impédance est implémenté de manière similaire aux bras du HRP2 des travaux de [Takubo *et al.*, 2005]. Pour rester en équilibre en appliquant une force sur l'objet, la trajectoire du COM est utilisée avec celle du ZMP pour recalculer en temps réel la position du robot à partir d'un modèle analytique, selon chacune des phases de mouvement. Afin d'éviter une discontinuité dans la vitesse au moment de connecter la nouvelle et l'ancienne trajectoire, le ZMP doit aussi être recalculé avec les conditions initiales du COM.

Malgré les différences d'implémentations, plusieurs concepts et méthodes de contrôle restent les mêmes. Entre autres, puisque le HRP2 possède des capteurs de forces/couples dans

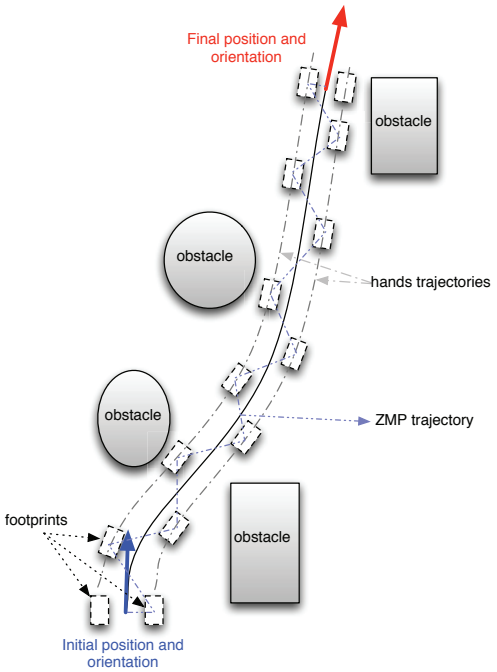
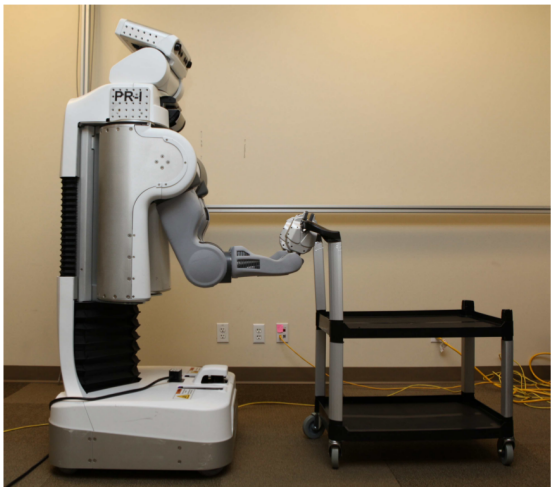


Figure 2.2 Exemple d'utilisation de trajectoire du ZMP pour gérer l'équilibre d'un robot.

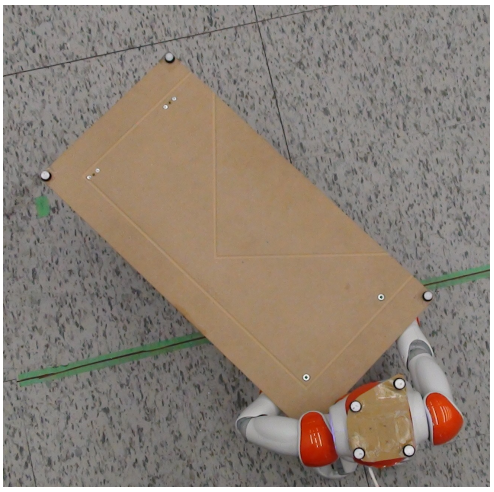
les mains, le contact avec l'objet ainsi que l'interaction entre celui-ci et les mains du robot est généralement effectué par un contrôle par impédance. Afin de pouvoir implémenter ce type de contrôle, il est nécessaire d'avoir ces informations. Par contre, la plupart des robots humanoïdes à faible coût ne possèdent pas ce genre de capteurs plutôt dispendieux, ce qui rend cette technique non généralisable. Également, pour pousser l'objet avec un humanoïde, la modification du ZMP par une méthode spécifique est populaire. Cette façon de procéder est efficace pour un problème en boucle ouverte, tel que pousser. En effet, une modification du ZMP en boucle ouverte ne crée des forces que sur l'objet à déplacer, ce qui est désiré pour générer du mouvement. Cependant, dans une coopération entre deux robots en boucle fermée, ces forces vont se propager jusqu'à l'autre robot, créant des interactions indésirables et déstabilise le système.

Tous les travaux présentés précédemment comportent un problème commun important : aucun d'entre eux n'utilise le contrôle des mains afin de pouvoir tourner ou bouger l'objet par rapport au robot afin d'optimiser son déplacement et son empreinte. En effet, les robots et leurs objets sont généralement déplacés comme étant un seul bloc rigide bougeant sur un plan. Les travaux de [Scholz *et al.*, 2011] faits sur le PR2 par contre utilisent cette possibilité afin de contrôler la position d'un petit chariot. Bien que le PR2, illustré à la figure 2.3 a), ne soit pas humanoïde, la recherche de solution dans un espace de configura-

tion considérant la rotation du chariot par les mains permet de non seulement optimiser l'espace de déplacement du chariot, mais également de faire des virages très serrés qui ne seraient pas possibles sans cela. La figure 2.3 b) montre un robot Nao tournant une table selon ce principe jusqu'à un angle de 30°. Plusieurs centaines de primitives représentent tous les mouvements possibles du PR2, puis elles sont utilisées par un algorithme de planification de mouvement par échantillonnage qui teste toutes les possibilités. Par contre, le PR2 a l'avantage d'être sur une base mobile omnidirectionnelle, ce qui permet d'éviter les perturbations latérales et verticales lors du mouvement. Également, il peut plus aisément suivre une trajectoire au sol qu'un robot humanoïde le peut avec des pieds. Finalement, puisqu'il n'y a qu'un seul robot, il s'agit d'une chaine cinématique ouverte plutôt qu'une chaine cinématique fermée. Un retour de force causé par une coopération n'affecte donc pas ses performances.



(a) PR2 tenant un petit chariot [Scholz *et al.*, 2011].



(b) Nao tournant une table miniature au maximum.

Figure 2.3 Utilisation des bras pour déplacer un objet.

Une alternative à pousser un objet sur le sol est de soulever l'objet du sol pour le transporter dans les mains ou encore les bras du robot. C'est d'ailleurs grâce à une peau artificielle que [Ohmura et Kuniyoshi, 2007] ont pu soulever une charge de 30 kg dans les bras de leur robot humanoïde. Pendant les expériences, une multitude d'informations est tirée des capteurs tactiles. Le robot se tient sur une table en poussant avec un bras, tandis que de l'autre il atteint l'objet. Il détermine ensuite les dimensions de celui-ci avec ses deux bras et l'apporte à son torse pour valider sa position. Une fois posée sur les avant-bras, la peau permet de connaître exactement l'emplacement de l'objet ainsi que la répartition de son poids. Toutes ces informations mises ensemble permettent un contrôle très précis d'une charge très lourde qui est beaucoup plus élevée que ce qui est possible lorsque portés à

bout de bras. En effet, en mettant la charge plus près des moteurs, le couple appliqué sur l'objet est beaucoup plus grand. De plus, le poids reste plus près du COM et donc génère moins de force et d'instabilité sur le robot. Par contre, les capteurs tactiles nécessaires sont très dispendieux et le temps de calcul qui doit être alloué à la lecture et l'analyse d'une aussi grande quantité de capteurs est particulièrement élevé.

Un autre exemple de robot humanoïde transportant un objet en le soulevant est un HRP2 qui déplace un poids de 8.5 kg [Harada *et al.*, 2005]. Sans utiliser une peau spéciale, se sont les capteurs de forces/couples dans ses mains et dans ses chevilles qui fournissent les données pour effectuer cette tâche tout en restant en équilibre. Le défi principal de ce travail est que le robot ne connait pas d'avance ni le poids, ni le centre de gravité de l'objet et doit donc les estimer avant d'ajuster son équilibre pour le soulever et marcher. Le robot commence donc par tenter de soulever l'objet en ligne droite. Cela lui permet de déterminer les informations précédemment manquantes pour ajuster l'équilibre durant sa marche, soit la masse et le centre de masse de l'objet. Ensuite, l'équation du ZMP est adaptée pour prendre en compte ces entrées supplémentaires et rester en équilibre. Les expériences démontrent que sans ces ajustements, le robot tombe vers l'avant (la direction de l'objet), à cause du poids élevé de l'objet.

2.4 Cooperation humanoïde-humanoïde

Le principe de contrôle en maître/esclave peut être également utilisé pour des interactions humanoïde-humanoïde. En effet, d'autres expériences ont été faites en utilisant cette fois deux HRP2, l'un en mode maître et l'autre en mode esclave [Wu *et al.*, 2011]. Encore une fois, les capteurs de forces dans les mains permettent au robot esclave de connaître la direction du maître, alors que le maître lui se concentre à suivre un chemin fourni directement par des commandes d'un opérateur humain externe. Ici, l'emphase est mise sur l'équilibre du robot esclave qui ne doit jamais tomber lorsque le maître bouge brusquement dans une direction. Par contre, les mouvements considérés ne sont que dans l'axe sagittal des robots. Bien que cette fois, il s'agit vraiment de deux robots humanoïdes tenant un objet, et non pas un humain et un robot, les problèmes du mode maître/esclaves pointés précédemment sont les mêmes, car il s'agit du même robot HRP2.

Étant donné que tous les robots ne sont pas dotés de capteurs de force dans les mains, [Inoue *et al.*, 2007] ont développé un système maître/esclave se basant plutôt sur l'information visuelle prise par les caméras de robots humanoïdes de petite taille et très peu dispendieux. Le HOAP-1 en mode esclave utilise trois critères visuels afin de déterminer

où se situe le maître par rapport à lui-même dans l'espace et le suivre. Ces trois critères illustrés à la figure 2.4 sont : la grosseur du robot qui détermine la distance, la distance par rapport au centre de l'image qui détermine son décalage latéral et son angle d'inclinaison qui détermine sa rotation autour de l'axe vertical. Le choix de la primitive à exécuter dépend de l'ensemble de ces facteurs. Un algorithme d'apprentissage permet d'adapter le choix des primitives selon les expérimentations. L'utilisation d'une caméra au lieu des capteurs de force est beaucoup plus versatile et pratique, car la grande majorité des robots humanoïdes ont une ou plusieurs caméras dans la tête pour effectuer différentes tâches. Par contre, les résultats indiquent que l'utilisation d'une caméra donne des valeurs imprécises et qu'il est très difficile de réaliser ce système de manière fiable.

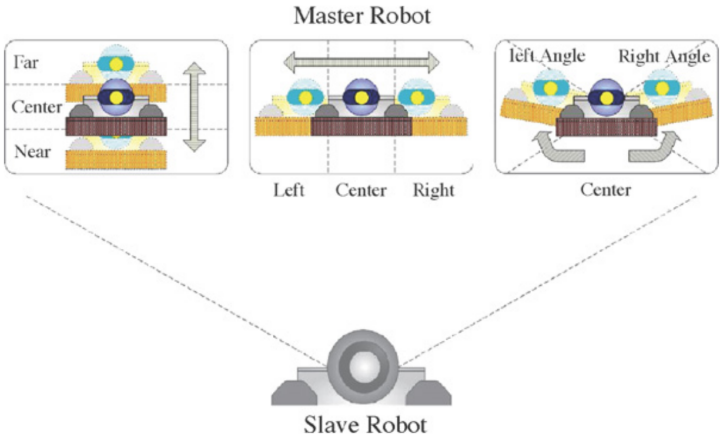


Figure 2.4 Critères visuels perçus par le robot esclave [Inoue *et al.*, 2007].

Il n'est pas obligatoire de se fier uniquement à l'information des caméras du robot pour s'en servir de manière efficace dans une coopération humanoïde. En effet, dans un système comportant deux robots DARwIn-OP, le but a été de repérer l'objet à transporter, s'y diriger pour le prendre, puis se déplacer avec [McGill et Lee, 2011]. La prise de l'objet, son lever ainsi que le déplacement avec celui-ci se fait de façon synchronisée tout au long du processus. Les DARwIn-OP utilisent une caméra située dans leur tête pour détecter l'objet, mais également les yeux de l'autre robot afin de garder une distance et un angle particulier à chaque étape. Le modèle utilisé pour la marche a été simplifié à celui d'un quadrupède plutôt que deux bipèdes reliés par un objet. Bien que cette méthode permet de simplifier la marche, elle retire aussi beaucoup de liberté de mouvement en ajoutant des contraintes virtuelles. L'ensemble robot-objet-robot est alors considéré comme un rectangle rigide plutôt que comme trois blocs indépendants, encore une fois, simplifiant le problème en retirant de la liberté de mouvement.

Bien que dans la plupart des cas, l'objet à transporter se situe dans les mains du robot, il est possible de transporter un lourd et large objet en le tenant d'un manière différente, par exemple en le soutenant sur les épaules. Dans le but de reproduire un transport mobile traditionnel japonais du nom de Mikoshi, où plusieurs personnes soutiennent une plateforme sur laquelle une autre est assise, quatre robots Nao en portent un cinquième sur leurs épaules [Yoshikai *et al.*, 2012]. La structure résultante est montrée à la figure 2.5. Les informations sont envoyées vers un serveur externe pour y faire le traitement, puis sont relayées aux porteurs. La marche de ceux-ci est synchronisée afin que chacun fasse un pas en même temps et de la même façon. L'orientation de leurs bras est utilisée pour maintenir la structure en équilibre durant la marche. La synchronisation de la marche des porteurs permet à la structure de rester en équilibre, mais implique que l'oscillation latérale y sera appliquée directement. En d'autres mots, l'objet porté oscille grandement lui aussi d'un côté à l'autre à chaque pas, et il faut généralement éviter ce facteur d'instabilité non négligeable.

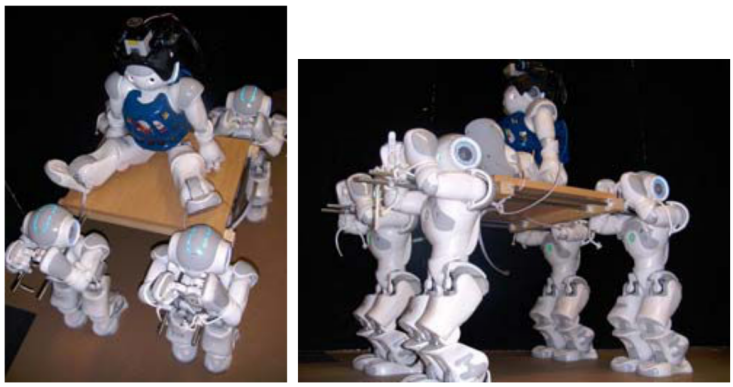


Figure 2.5 Quatre Nao en transportant un cinquième sur un Mikoshi [Yoshikai *et al.*, 2012].

Dans le travail de [Yoshikai *et al.*, 2012], le robot positionné sur la plateforme possédait tous les capteurs, lasers et caméras nécessaires pour effectuer la localisation de l'ensemble du Mikoshi et donc gérer la navigation dans l'environnement encombré. Il est primordial d'avoir une localisation précise qui corrige l'odométrie, car la contre réaction des joints et le glissement des pieds au sol la rendent rapidement divergente et inexacte. Cependant, il est difficile de faire cette correction pour un robot humanoïde, car celui-ci bouge constamment latéralement en se déplaçant. [Hornung *et al.*, 2010] utilisent également un capteur de distance laser pour faire la correction de l'odométrie du Nao dans l'espace et obtenir une meilleur localisation. En effet, à l'aide d'un capteur laser positionné au-dessus de la tête du robot, les données sont fusionnées aux informations de roulis, de tangage et de hauteur fournies par l'odométrie avec un algorithme de Monte Carlo Localization (MCL).

En lui fournissant l'ensemble de la carte de l'environnement, MCL permet de déterminer précisément la position et l'orientation actuelles du robot à l'aide des informations des capteurs présentes et passées. Cela signifie par contre qu'il faut avoir un modèle complet en 3D de l'environnement où se déplace le robot sous forme de carte pouvant être utilisée par l'algorithme. Également, étant donné que le laser se trouve sur la tête du Nao, l'utilisation d'une table et d'un autre robot pour coopérer bloque la vue du sol et nuit grandement au bon fonctionnement du capteur.

2.5 Vision

Puisque le Nao dispose de des caméras dans la tête, plutôt que d'ajouter des capteurs supplémentaires, elles peuvent être utilisées afin de faire la localisation précise dans l'environnement [Oßwald *et al.*, 2010]. Dans leur travail, l'algorithme d'apprentissage nommé Markov Decision Process (MDP) est utilisé pour optimiser la détection de caractéristiques particulières dans l'environnement. L'objectif de cet algorithme est de maximiser la récompense obtenue pour une série d'actions A entre des états S. La récompense est réduite avec le temps de manière à favoriser les parcours les plus rapides. Les actions A choisies sont : avancer de 10 cm, tourner de 23 degrés et s'arrêter 0.7 secondes afin d'obtenir une image claire pour en extraire les caractéristiques. L'état S est : la distance au but, l'angle relatif au but et l'entropie de la localisation. Bien que ce système permet d'obtenir une localisation plus robuste qu'avec seulement les données d'odométrie, la vitesse de marche est réduite pour limiter le flou de mouvement dans l'image qui rend l'identification de caractéristiques dans l'image difficile. Pour cette même raison, le robot doit prendre des pauses et s'arrêter pour obtenir une image claire lorsque l'entropie devient trop élevée. De plus, l'ensemble des actions est limité à trois primitives de base, dont une est un arrêt, ce qui limite énormément les mouvements du robot. Finalement, puisque la caméra utilise principalement les caractéristiques du sol pour se localiser, un objet transporté positionné devant le robot rend cette technique inutilisable.

En utilisant un HRP2 et une seule caméra à angle de vue large, [Stasse *et al.*, 2006] ont réalisé un SLAM (Simultaneous Localization and Mapping). Leur technique de détection de caractéristiques leur a permis de pratiquement éliminer l'erreur d'odométrie du robot lorsque celui-ci effectue une trajectoire circulaire courte. Cependant, afin d'obtenir des résultats adéquats, il leur a été nécessaire d'ajouter des repères artificiels dans l'environnement, sans quoi l'algorithme ne parvenait pas à détecter suffisamment de caractéristiques. Également, afin de correctement corriger la localisation du robot, l'algorithme doit détecter des boucles, c'est à dire de revenir à une position où il est précédemment

passé, ce qui n'est pas le cas de plusieurs trajectoires lors d'un déplacement quelconque. De plus, puisque le système est embarqué en haut du robot, encore une fois, n'importe quel objet transporté devant le robot bloquera une grande partie du champ de vision du robot. Finalement, tandis que le HRP2 possède deux processeurs distincts, dont un dédié uniquement au traitement d'images tel que présenté ici, le Nao quant à lui est beaucoup plus restreint en puissance de calcul. Un exemple de leur algorithme est montré à la figure 2.6, avec à gauche la vue du robot et les caractéristiques particulières détectées dans l'environnement, et à droite le plan général de la localisation du robot ainsi que la position des caractéristiques détectées et leurs incertitudes.

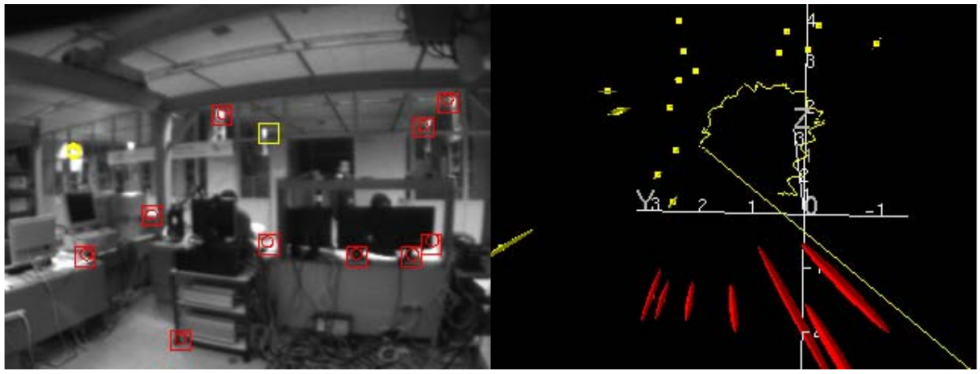


Figure 2.6 SLAM effectué par un HRP2 durant un mouvement circulaire, tiré de [Stasse *et al.*, 2006].

CHAPITRE 3

Navigation de robot humanoïde autonome dans un environnement encombré en transportant une charge lourde

Auteurs et affiliation :

A. Rioux : étudiant à la maîtrise en génie électrique, Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.

W. Suleiman : professeur, Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.

Date de soumission : 15 décembre 2015

Revue : Robotics and Autonomous Systems

Titre anglais : Humanoid Navigation and Heavy Load Transportation in a Cluttered Environment

Contribution au document : Nos contributions sont les suivantes : (1) un système de contrôle de tout le corps qui demande au robot humanoïde d'utiliser ses mains et ses bras pour contrôler les mouvements du système à chariot (par exemple, lors de virages serrés) ; (2) une approche sans capteur pour sélectionner automatiquement l'ensemble de primitives appropriées en fonction du poids de la charge ; (3) un algorithme de planification de mouvement qui génère une trajectoire sans collisions en utilisant le jeu approprié de primitives et la carte construite de l'environnement ; (4) une technique de filtrage efficace pour ignorer le chariot et la charge du champ de vue du robot tout en améliorant les performances générales des algorithmes de SLAM ; et (5) un processus continu et cohérent d'odométrie formé en fusionnant les informations visuelles et celles de l'odométrie du robot.

Résumé français : Bien que ces dernières années il y a eu quelques études visant à la navigation de robots dans des environnements encombrés, seulement quelques-unes d'entre elles ont abordé le problème de robots qui naviguent en déplaçant un objet volumineux ou lourds. Ceci est particulièrement utile pour faire le transport de marchandises avec des poids et des formes variables, sans avoir à modifier le robot physiquement. Inspiré

par l'utilisation répandue de chariots de fortune par les humains, nous nous intéressons, dans ce travail, au problème d'un robot humanoïde naviguant dans un environnement encombré tout en déplaçant une charge lourde qui se trouve sur un chariot de fortune. Nous présentons un système de navigation complet, de la construction incrémentale d'une carte de l'environnement et du calcul des trajectoires sans collision à la commande pour exécuter ces trajectoires.

Nous présentons des expériences menées sur un robot Nao, équipé d'un capteur RGB-D monté sur la tête, poussant un chariot avec différentes masses. Nos expériences montrent que la charge utile peut être significativement augmentée sans changer physiquement le robot, et en augmentant donc la capacité du robot humanoïde dans des situations réelles.

3.1 Abstract

Although in recent years there have been quite a few studies aimed at the navigation of robots in cluttered environments, few of them have addressed the problem of robots navigating while carrying large or heavy objects. This is especially useful when transporting loads with variable weights and shapes without having to change the robot's hardware. Inspired by the wide use of makeshift carts by humans, we tackle in this work the problem of a humanoid robot navigating in a cluttered environment while moving a heavy load lying on a cart-like object. We present a complete navigation scheme, from the incremental construction of a map of the environment and the computation of collision-free trajectories, to the control for executing these trajectories. Our contributions are as follows : (1) a whole-body control scheme that makes a humanoid robot use its hands and arms to control the motion of the cart-load system (e.g., in tight turns) ; (2) a sensorless approach to automatically select the appropriate primitive set according to the load weight ; (3) a motion planning algorithm to find an obstacle-free trajectory using the appropriate primitive set and the constructed map of the environment as an input ; (4) an efficient filtering technique to remove the cart from the field of view of the robot while improving the general performances of the SLAM algorithms ; and (5) a continuous and consistent odometry data generation formed by fusing the visual and the robot odometry information. We present experiments conducted on a real Nao robot, equipped with a RGB-D sensor mounted on its head, pushing a cart with different loads. Our experiments show that the payload can be significantly increased without changing the robot's main hardware, and therefore enacting the capacity of humanoid robots in real-life situations.

3.2 Introduction

One of the advantages of having arms on a robot is that it can carry a load. This capacity can be useful for a wide range of actions, including transporting objects from one place to another. However, the maximum payload is generally pretty low and generates a lot of instability when held at arm's length. While it is possible to increase the strength of the motors in the legs and arms, it is not the best solution since motor power is proportional to its size, weight and price. Instead of putting the entire load directly on the robot, a cart-like object can be used to help support the weight. The structure can then be pushed by the robot and moved around more easily without having to modify the robot's hardware to fit the load.

Many studies have been done on robots moving objects to a specific goal. Most of them though are executed with multiple wheeled robots that position themselves around the object to push it in the desired direction [Kube et Bonabeau, 2000; Miyata *et al.*, 1997; Ota *et al.*, 1995; Wang *et al.*, 1999; Yamashita *et al.*, 2003]. In these works, the manipulator comes in contact with the object at only one point, which barely allows full control of the object while moving and manipulating it. Holonomic wheeled robots are less complex to control than humanoid robots and are mainly used here to slide box-like object on the ground. Robots with humanoid arms have a better control on the structure of an object and may be therefore more suitable to control objects with various shapes.

More advanced and specialized models of wheeled robots can possess humanoid torsos and arms which give them the same capacity to keep a high level of control when holding or transporting objects [Hirata et Kosuge, 2000] [Suda et Kosuge, 2002] [Lawitzky *et al.*, 2010]. However, most environments, made by and for humans, are more suitable for humanoid robots. To this end, the subject of bulky or heavy objects transport with a humanoid robot has received attention in the past years. To achieve this task, many different techniques have been developed. For instance, lifting the load from the ground [Harada *et al.*, 2005] [Ohmura et Kuniyoshi, 2007] [Yoshida *et al.*, 2008] or using a part of the transported object as a pivot to move it [Aiyama *et al.*, 1993] [Yoshida *et al.*, 2006] [Yoshida *et al.*, 2010].

Other works have explored the usage of humanoid robots that push heavy objects that are placed on a cart while keeping a firm grip on the handles. In theory, two arms are enough to fully constrain and control all the degrees of freedom of a cart. It was demonstrated that Honda's ASIMO is capable of moving a large cart in rooms and hallways [Shigemi *et al.*, 2006] and a HRP-2 [Kaneko *et al.*, 2004] is able to push a person on a wheelchair [Nozawa *et al.*, 2008]. However, in both cases, the cart is mostly controlled as a Dubins car [Dubins, 1957] instead of taking advantage of the full possibility of the humanoid robot's holonomic movement. Also, the arms serve only as a mean for attaching the cart to the robot and are not taken into consideration to control the cart further.

In [Stilman et Kuffner, 2004], a planning method for humanoids to navigate among movable obstacles has been proposed. The main purpose of that method is to find a path from a starting point to a goal point in a complex environment where the robot can easily move objects to create a clear path, if it exists. Our objective is however different, as we are interested in not only navigating in a cluttered environment, but also transporting a heavy load that is significantly bigger than the humanoid payload.

In the work of [Scholz *et al.*, 2011], a PR2 robot possessing a wheeled holonomic base and two 7 DoF (Degree of Freedom) arms is used to push a small cart and control its orientation. However, despite having humanoid arms to orient the cart, since the PR2 has wheels instead of legs, no lateral swing is transmitted to the transported object, causing oscillations and instability, which is a problem with humanoid robots. Furthermore, the cart is very small and light weight with respect to the PR2, in contrast to our proportionally bigger cart that is able to carry a load heavier than the robot itself.

The main contribution of our work is on providing a complete framework for autonomous humanoid robot navigation in a cluttered environment while manipulating a cart-like object that carries a heavy load. In addition, an efficient swing reduction control algorithm and an automated sensorless primitive set selection based on the load weight are explained. Finally, an optimized algorithm that filters out the cart's point cloud and an improved odometry data are proposed.

To address the problem of navigating a cart-like object supporting a load with a humanoid robot in a cluttered environment, the following sub-problems should be solved : (I) planning ; (II) controlling the cart-like object ; and (III) sensing the environment. This paper is organized according to these sub-problems. Section 3.3 presents an anytime search-based planner that exploits a given set of motion primitives, which consider both the robot and the cart footprint in order to plan a safe trajectory between obstacles. Section 3.4 describes how to find the humanoid robot's footprints and then compute the humanoid's feet and hands trajectories in order to minimize the swing effect. It also describes how to follow the cart trajectory using a task priority whole-body control scheme. In Section 3.5, real-time information from an entry level depth camera is used to perform simultaneous localization and mapping (SLAM) of the surrounding obstacles in the cluttered environment. Finally, in Section 3.6, results of simulation and real world experiments are presented and analyzed.

3.3 Planning a Valid Path

To be able to navigate through a cluttered environment, a path provided by a motion planning algorithm is essential. Among the different possibilities, we chose a lattice-based graph planning with an ARA* search [Likhachev *et al.*, 2004]. This choice is mainly motivated by the use of motion primitives that assure feasible robot-cart configurations and transitions. The environment is modeled by a 2D grid costmap that discriminates obstacles from free space at a fixed threshold, and allows obstacles inflations to increase the security margin.

3.3.1 State Representation

Each node of the search graph needs a complete state representation of the robot-cart to properly operate. To achieve this, it is possible to model the state in $\mathbb{R}^2 \times \mathbb{S}^1 \times \mathbb{R}^2 \times \mathbb{S}^1$:

$$s = (x_r, y_r, \theta_r, x_{ca}, y_{ca}, \theta_{ca}) \tag{3.1}$$

where x_r, y_r and θ_r are the positions and orientation of the robot, and x_{ca}, y_{ca} and θ_{ca} are those of the cart. The problem can be however simplified by setting a pivot point positioned in the middle of both hands, to reduce the dimensionality of the search space. This simplification is valid in our case because : I) the working space of our robot's arms is too small to fully take advantage of both rotation and translation ; II) we consider that the robot is always grasping the cart handles during the execution of the trajectory. The closed loop grasping of the robot on the table is shown in Fig. 3.1. The chosen position of the pivot point maximizes the rotation range within the robot's workspace, resulting in a 4-dimension state space $\mathbb{R}^2 \times \mathbb{S}^1 \times \mathbb{S}^1$:

$$s = (x_r, y_r, \theta_r, \theta_{ca}) \tag{3.2}$$

Even though the above simplification removes the ability of the cart to translate on the plane, the robot retains enough manipulability to minimize the cart footprint on tight turns. The simplified state representation of equation (3.2) is shown in Fig. 3.1.

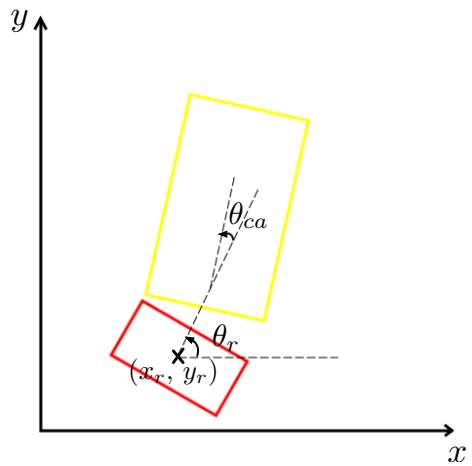


Figure 3.1 Simplified state representation.

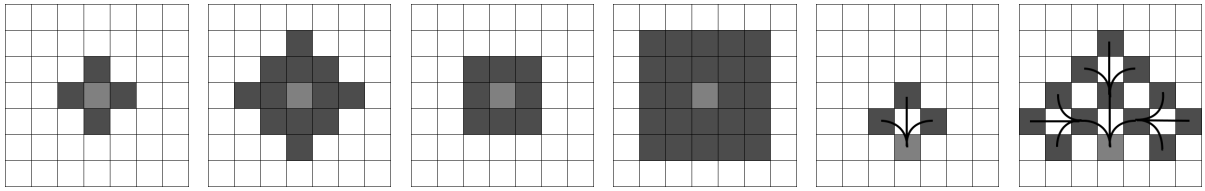


Figure 3.2 Different forms of graph search methods. From left to right : Von Neumann neighborhood 1st & 2nd expansions, Moore neighborhood 1st & 2nd expansion, example of primitives set 1st & 2nd expansions.

3.3.2 Transition Model

In a lattice-based graph planner, the transition between the nodes is a discrete action chosen within a fixed-set of possible actions called motion primitives. Motion primitives allow the decomposition of complex motion generation in robotics and it is very likely that such arrangement of atomic motions are also used by humans and animals [Flash et Hochner, 2005; Hart et Giszter, 2010]. An important feature of the lattice representation is that each of those connections is a feasible path, in contrast to other commonly used forms of graph search, including Von Neumann neighborhood [Weisstein, 2012] or Moore neighborhood [Weisstein, 2005]. An example of the first two expands of these graph search methods is shown at Fig. 3.2. This makes it really suitable for highly constrained systems, such as a robot moving a cart.

Because the cart can carry different loads, multiple sets of primitives are needed depending on the load weight. Without load, the robot is holonomic and can move in any direction. A subset of movements composed of forward, backward, diagonal, rotate in place and turn while moving forward is used to reduce planning time while focusing on forward movements. With a heavy load though, moving sideways and rotating in place becomes really difficult because of the increased friction. For this reason, when the weight becomes too important, the robot's rotation occurs around a pivot point situated between the two table legs touching the ground. Thus, changing the feasible primitives is necessary for the lattice representation to remain coherent. Figure 3.3 presents an example of right turns for the omnidirectional and the heavy load sets of primitives.

3.3.3 Automatic Selection of Primitive Sets

To determine which set of primitives is the most appropriate to be used, an estimation of the load weight is necessary. The easiest solution to automatically determine this value is by lying multiple pressure sensors on the table surface. However, this solution is impractical

(a) omnidirectional primitives set

(b) heavy load primitives set

Figure 3.3 A right turn executed by the robot (blue) pushing the cart (green) to the goal (red arrow) with both sets of primitives.

and requires adding external hardware to the robot. Instead, path following trials are performed by the robot upon startup. Indeed, since the friction creates a pivot point at high weight, the proper motions of the robot are hindered. For instance, if a command to turn in place, walk sideways or turn the table is sent, it won't be possible for the robot to correctly execute it. As a result, we expect that the error between the internally computed motion of the robot and the visually perceived one would increase proportionally to the load weight.

Depending on the robot, table and load, multiple basic motion candidates can be used and need to be evaluated.

- Walking forward in the sagittal plane (X direction), because greater loads should cause proportionally more feet slippage.
- Walking in the lateral direction (Y direction), because greater loads should cause the robot to proportionally turn around the wheels pivot instead of walking straight.
- Turning in place, because greater loads should proportionally reduce the angular rotation.

Then, to be able to use this movement error, it is necessary to have at least one threshold to separate the categories. For the time being, there are only two categories : the first with zero to low weight that does not impair too much the movement, and the second with heavy load that highly restricts the possible motions. Therefore, we have the following two hypothesis :

1. Hypothesis H_0 : the carried load weight is low and the most appropriate primitive set is formed by omnidirectional primitives.

2. Hypothesis H_1 : the carried load weight is heavy and the most appropriate primitive set is formed by heavy load primitives.

Let us suppose that the probability distribution functions of H_0 and H_1 are Pr_{H_0} and Pr_{H_1}, respectively. A threshold x that minimizes the probability of false detection of the correct hypothesis can be defined as the point at which Pr_{H_0} and Pr_{H_1} are equal :

$$Pr_{H_0}(\mathcal{X} = x) = Pr_{H_1}(\mathcal{X} = x) \tag{3.3}$$

where $\mathcal{X}$ is a random variable. Figure 3.4 gives an example of the application of this method.

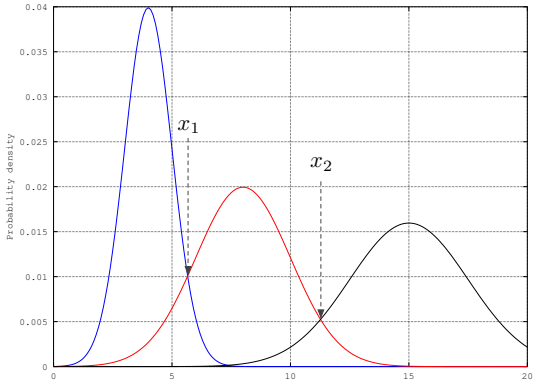


Figure 3.4 Thresholds selection in the case of three normal probability distribution functions.

With enough samples for each primitive set, a normal distribution can be used to evaluate their probabilities. Using this method, it is easy to find $N-1$ thresholds in the case of N primitives sets. However, since every distribution gets closer to each other and starts to overlap as N increases, the probability of false detection also increases.

3.3.4 Path Cost Function

The cost function of a transition, from state s to s' is based on the time to execute that transition and is computed as follows :

$$g(s,s') = \begin{cases} \frac{\sqrt{(\Delta x_r)^2+(\Delta y_r)^2}}{\dot{r}^+} \times DF & \text{if } \Delta x_r \neq 0 \text{ or } \Delta y_r \neq 0 \\ \frac{\Delta \theta_r}{\dot{\theta}^+} \times DF & \text{otherwise} \end{cases} \tag{3.4}$$

where Δx_r, Δy_r and $\Delta\theta_r$ are the differences between the x_r, y_r and θ_r, which are the coordinates of the robot's pelvis joint, between states s and s', DF is a difficulty factor associated with each primitive, $\dot{\boldsymbol{r}}^+$ is the maximal robot linear velocity and $\dot{\theta}^+$ is the maximal angular velocity for turning in place. The Euclidean distance between both states is computed and then divided by the maximum velocity of the robot in the direction of the movement to give the approximate time to execute the primitive. Since the robot usually slows down when approaching an obstacle to avoid collision, transitions that pass close to obstacles have higher costs.

For both instances, the time cost estimate is then multiplied by the DF associated with each primitive. This DF is used to prioritize or penalize certain motions or directions, which result in a smoother and a more natural looking trajectory. For instance, turning in place then moving forward takes a longer time than moving in diagonal. However, on a long distance, the former reduces the trajectory footprint and is therefore more natural looking while reducing the chances of drifts caused by the table movements. For those reasons, moving sideway has a higher DF than turning and moving forward. To sum up, for small distances, the time cost takes over the DF, however for long distance, it is more likely that turning in place and moving forward would be preferred. Table 3.1 presents examples of DF values.

Tableau 3.1 The DF factor for different motion classes of each set

	Omnidirectional set	Heavy set
Forward	1	1
Turn in place	2	2
Backward	3	N/A
Sideways	2	N/A
Diagonal	1	N/A

3.3.5 Search Algorithm

A* is one of the most popular search algorithm to find an optimal solution path using a cost function. In addition to the path cost function, a heuristic bias the search towards the most promising states. Even though A* is optimal when it finds a solution, that solution may, however, not always exist or cannot be found within a reasonable time. The Anytime Repairing A* (ARA*) planner focuses on delivering a suboptimal solution as fast as possible. This solution is then optimized iteratively to obtain the optimal solution within a predefined limited time. Also, the states are expanded in the opposite way, from goal to start, so that the heuristic costs remain valid after replanning and do not need to be

recomputed. Since the algorithm is a time constrained sub-optimal A*, it also guarantees completeness. The cost function takes the form of :

$$f(s, s') = g(s, s') * max(Cost_{cells}(s, s')) + \epsilon h, \epsilon >= 1 \tag{3.5}$$

where $g(s, s')$ is the path cost of Equation (3.4), h the heuristics that uses a grid of 2D distance cost computed with a Dijkstra search from the goal to the start states, and :

$$Cost_{cells}(s, s') = \begin{cases} 1 & \text{free space} \\ 2 \text{ to } 99 & \text{inflation} \\ \infty & \text{obstacles} \end{cases} \tag{3.6}$$

is a vector containing the cost of every cell between s and s'. The search is biased towards states that are closer to the goal and returns a solution that is, at worst, ϵ times the cost of the optimal solution, providing user defined bounds on the sub-optimality of the solution.

3.4 Controlling the Robot and Cart-Like Object

A collision free optimal path on the ground is not sufficient to safely control our humanoid robot and its object. In order to do so, the objects and the hands are stabilized, and a whole-body control scheme is used to transform the path into actual robot motions.

3.4.1 Object Stability and the Hand Stabilization

Since the cart is fully controlled by the robot's hands, it is possible to reduce the lateral swing created by the humanoid walk. In addition, the load stability is improved by restraining the propagation of these oscillations to the table. At low speed, a humanoid robot has no other choice than to move the horizontal projection of its Center of Mass (CoM) from one support (foot) to the other, in order to keep its Zero Moment Point (ZMP) within the support polygon and stay in balance. This lateral motion causes the entire upper body of the robot to oscillate laterally at an amplitude proportional to the distance between the center of the feet, which in turn causes the side of the cart, which is held by the robot, to move by the same amplitude. It is usually unsafe to transport a load that continuously swing from one side to another, and could even damage the transported objects or the surroundings.

One way to compensate this instability without changing the walking gait and feet position is to use the robot's arms. To this end, the hands are kept at a fixed position with respect to the fixed frame of the support foot. This position is determined at the initial starting position of the robot while holding the table and corresponds to the transformation between both feet and both hands. While moving, the transformation of the hands w.r.t. the fixed foot is used. Those transformations are the input of the whole-body control scheme described in the Section 3.4.2.

3.4.2 Whole-body Control Scheme

Once a collision-free trajectory is found by the ARA* algorithm, a set of footprints are defined along the trajectory as shown in Fig. 3.5. The second step is to set a dynamically stable trajectory by defining an appropriate ZMP trajectory [Kajita *et al.*, 2003]. A trajectory of the CoM of the robot is then obtained using the preview control algorithm proposed in [Kajita *et al.*, 2003]. This algorithm has been widely used by researchers in humanoid robotics. It is simple to implement, yet efficient and yields a smooth CoM trajectory by minimizing the CoM jerk trajectory. The feet trajectories are obtained by spline interpolation between the footprints and the hands trajectories, and orientations are defined to minimize the walking swing effect as well as follow the cart orientation.

To obtain the humanoid robot's joint trajectories, a whole-body control scheme with prioritized tasks is formulated as follows :

$$
\begin{aligned}
& \min_{\dot{\boldsymbol{q}}} \dot{\boldsymbol{q}}^T \boldsymbol{Q} \dot{\boldsymbol{q}} \\
& \text{subject to} \\
& \text{First priority} \quad \begin{cases} \boldsymbol{J}_c \, \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_c \\ \boldsymbol{J}_{lf} \, \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_{lf} \\ \boldsymbol{J}_{rf} \, \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_{rf} \end{cases} \\
& \text{Second priority} \quad \begin{cases} \boldsymbol{J}_{lh} \, \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_{lh} \\ \boldsymbol{J}_{rh} \, \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_{rh} \end{cases} \\
& \text{Joint velocity limits} \quad \hat{\dot{\boldsymbol{q}}}^- \leq \dot{\boldsymbol{q}} \leq \hat{\dot{\boldsymbol{q}}}^+
\end{aligned}
\tag{3.7}
$$

where $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ is the joint velocity vector, $\boldsymbol{Q}$ is a positive semi-definite matrix, $\boldsymbol{J}_c \in \mathbb{R}^{3\times n}$, $\boldsymbol{J}_{lf} \in \mathbb{R}^{6\times n}$, $\boldsymbol{J}_{rf} \in \mathbb{R}^{6\times n}$, $\boldsymbol{J}_{lh} \in \mathbb{R}^{6\times n}$, $\boldsymbol{J}_{rh} \in \mathbb{R}^{6\times n}$ are the jacobian matrices of the CoM, left foot, right foot, left hand and right hand respectively. $\dot{\boldsymbol{r}}_c$, $\dot{\boldsymbol{r}}_{lf}$, $\dot{\boldsymbol{r}}_{rf}$, $\dot{\boldsymbol{r}}_{lh}$, $\dot{\boldsymbol{r}}_{rh}$ are

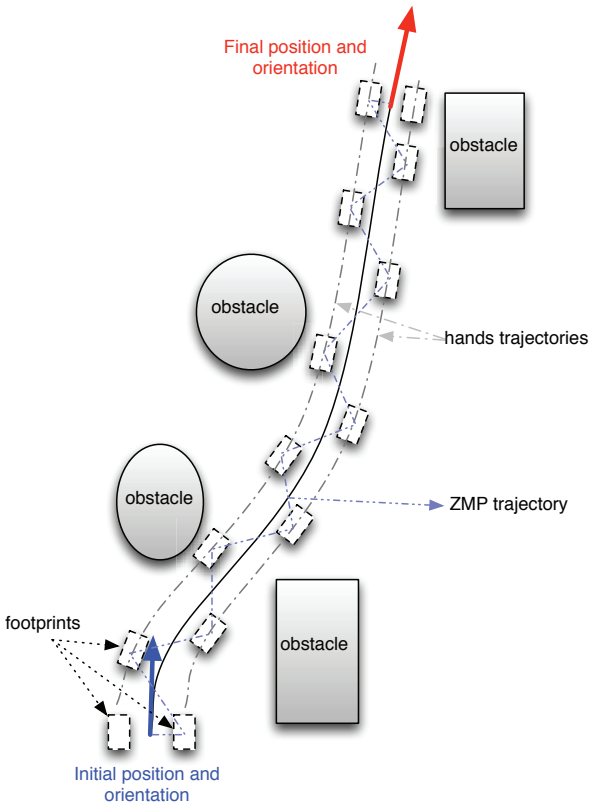


Figure 3.5 Overview of the motion planning procedure.

the linear and angular velocity of the CoM, left foot, right foot, left hand and right hand, respectively.

$\hat{\dot{\boldsymbol{q}}}^-$ and $\hat{\dot{\boldsymbol{q}}}^+$ are generalized joint velocity limits defined as follows :

$$\hat{\dot{\boldsymbol{q}}}_j^+ = \begin{cases} \dot{q}_j^+ \frac{(q_j^+ - q_j) - \rho_s}{\rho_i - \rho_s} & \text{if } q_j^+ - q_j \leq \rho_i \\ \dot{q}_j^+ & \text{otherwise} \end{cases}$$
$$\hat{\dot{\boldsymbol{q}}}_j^- = \begin{cases} \dot{q}_j^- \frac{(q_j - q_j^-) - \rho_s}{\rho_i - \rho_s} & \text{if } q_j - q_j^- \leq \rho_i \\ \dot{q}_j^- & \text{otherwise} \end{cases} \tag{3.8}$$

where $\hat{\dot{\boldsymbol{q}}}_j$ is the jth element of the vector $\hat{\dot{\boldsymbol{q}}}$, q_j is the value of joint j, q_j^+ and q_j^- are the upper and lower limits for the joint j, ρ_i and ρ_s are user-defined positive constants, ρ_i and ρ_s are usually called the influence and security distances, respectively. It can be easily proven that the equalities constraints in (3.8) not only yield a motion within the

humanoid's velocity limits, but also the joints limits are satisfied as well with a safety margin equals to ρ_s :

$$q_j^- + \rho_s \leq q_j \leq q_j^+ - \rho_s \tag{3.9}$$

Equation (3.8) provides a compact and efficient way to deal with both of velocity and joint limits, it has been originally proposed in [Kanehiro *et al.*, 2008].

The optimization problem (3.7) can be transformed into a standard Quadratic Programming (QP) problem as follows :

$$\begin{aligned} &\min_{\dot{\boldsymbol{q}},w} \dot{\boldsymbol{q}}^T \boldsymbol{Q} \dot{\boldsymbol{q}} + w^T \boldsymbol{Q}_w w \\ &\text{subject to} \\ &\boldsymbol{J}_1 \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_1 \\ &\boldsymbol{J}_2 \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_2 + w \\ &\hat{\dot{\boldsymbol{q}}}^- \leq \dot{\boldsymbol{q}} \leq \hat{\dot{\boldsymbol{q}}}^+ \end{aligned} \tag{3.10}$$

with :

- $$\boldsymbol{J}_1 = \begin{bmatrix} \boldsymbol{J}_c \\ \boldsymbol{J}_{lf} \\ \boldsymbol{J}_{rf} \end{bmatrix}. \tag{3.11}$$

- $$\boldsymbol{J}_2 = \begin{bmatrix} \boldsymbol{J}_{lh} \\ \boldsymbol{J}_{rh} \end{bmatrix}. \tag{3.12}$$

- $$\dot{\boldsymbol{r}}_1 = \begin{bmatrix} \dot{\boldsymbol{r}}_c \\ \dot{\boldsymbol{r}}_{lf} \\ \dot{\boldsymbol{r}}_{rf} \end{bmatrix} \text{ and } \dot{\boldsymbol{r}}_2 = \begin{bmatrix} \dot{\boldsymbol{r}}_{lh} \\ \dot{\boldsymbol{r}}_{rh} \end{bmatrix}. \tag{3.13}$$

where :

- w is a slack variable that is introduced to ensure the priority feature of (3.7).

- $\boldsymbol{Q}_w \in \mathbb{R}^{12\times12}$ is a user-defined positive-definite matrix. In order to respect the task priority, the following condition should be satisfied : $\|\boldsymbol{Q}_w\| \gg \|\boldsymbol{Q}\|$ (in practice, we usually use $\boldsymbol{Q}_w = 10^n \times \boldsymbol{Q}$, where n could be 3, 4 or 5).

The QP problem (3.10) can be solved in real-time by using an appropriate QP solver such as uQuadProg solver [Kanehiro *et al.*, 2010; uQuadProg Solver, 2009] or qpOASES solver [Ferreau *et al.*, 2014]. Note that the above formulation supposes that the first priority kinematics task is always feasible, and this is mainly true in our case. However, in a general case, a second slack variable should be added to the first priority constraints [Escande *et al.*, 2014]. The output of the above QP problem, $\dot{\boldsymbol{q}}$, is integrated to obtain $\boldsymbol{q}$ and then used as the desired reference for the robot's joints.

3.5 Simultaneous Localization and Mapping (SLAM)

To move in a cluttered environment, a robust and precise sensing input is primordial to determine the position of obstacles, detect collisions and to plan valid long term and short term paths. Also, odometry drift must be constantly verified and corrected by an accurate localization mechanism to ensure that the planned path is closely followed. The human-size humanoid robots, such as HRP-2 or the humanoids robots which participated in DARPA Challenge [Defense Advanced Research Projects Agency, 2016], are able to build a 3D map on the fly using their very sophisticated proprioceptive and exteroceptive sensors. However, the Nao robot only has two cameras in its head for sensing and localization. A first approach would be using those cameras, however this has proven to be a very difficult task [Stasse *et al.*, 2006] [Oßwald *et al.*, 2010]. Indeed, since a humanoid robot swings laterally while walking, and this effect coupled with low resolution cameras leads to pictures of poor quality. Furthermore, the field of view of the Nao is greatly obstructed by the large table and load. As a result, it is hard to precisely determine the position of the environment and obstacles with respect to the robot while only relying on the Nao's cameras.

3.5.1 Real-Time Appearance-Based Mapping (RTAB-Map)

Another approach would be adding a depth camera on the top of Nao's head for mapping [Maier *et al.*, 2012]. To achieve this, an open-source library named RTAB-Map [Labbe et Michaud, 2014a,b] has been used. RTAB-Map is a RGB-D Graph SLAM library that uses a bag-of-words technique for loop closure detection. A memory management system

limits the quantity of information loaded in memory to ensure the constant satisfaction of large environment real time constraints. RTAB-Map also provides a robust odometry system based on visual information. It can create 3D maps of the environment as well as constructing a 2D occupancy grid map by projecting the obstacles on the ground plane. Even though planning algorithms can use probabilistic grids, the library only provides deterministic occupancy grids at the moment. The library also supports large maps with kilometers long paths, multi-sessions mapping and localization.

3.5.2 Filtering Out the Cart

Since the cart is in the field of view of the camera, its point cloud needs to be filtered out in order to construct a valid occupancy grid or it will constantly be treated as an obstacle. The easiest solution is to simply ignore the part of the image where the cart is located. However, filtering a fixed cart position does not take into account the arms that can turn the table and the oscillation of the camera with respect to the table, as shown in Fig. 3.6. Moreover, removing the entire zone where the cart might be located results in ignoring the majority of the image and leads to poor performances.

Algorithm 1 GetObstacles

```
flatSurfaces ← normalFiltering(groundNormalAngle)
if (useSegmentation) then
  clusteredSurfaces ← extractClusters(flatSurfaces)
  ground ← getBiggestCluster
  for all cluster ∈ clusteredSurfaces do
    if cluster.centroid.normal is close to ground.centroid.normal then
      ground += cluster
    end if
  end for
else
  ground ← flatSurfaces
end if
obstacles ← extractClusters( !ground)
```

An efficient solution is to label all 3D points lying on planes parallel to the ground as free space. Without filtering, it means that when useSegmentation is true in Algorithm 1, only the ground is extracted and considered as free space, and all the rest are obstacles. By taking into account that the vast majority of obstacles are bounded by edges that are not parallel to the ground, it is possible to filter out all flat surfaces with a normal parallel to the ground normal. This allows to filter the cart no matter its position and orientation, thus using the points on its sides for localization and mapping.

The edges of obstacles generate walls around them that prevent motions to be planned over the obstacles. Note that boxes, tables, desks and chairs with a non zero width and their legs will all be easily detected. Only very thin surfaces with no distinguishable width and visible legs will be wrongly ignored. We believe, however, that this type of objects is not representative of typical obstacles. This filtering technique also reduces the required calculations to search for the ground by the vision library, since the clustering of flat surfaces is bypassed.

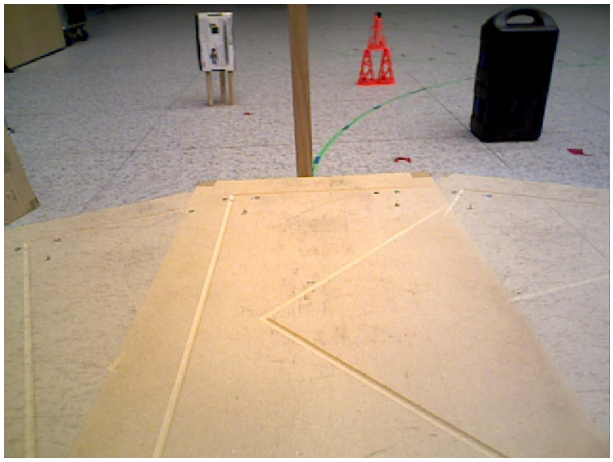


Figure 3.6 The position of the cart as seen by the robot. In transparency, the position of the cart when being turned at maximum angle.

3.5.3 Odometry Fusion

A problem that occurs when using the visual odometry produced by RTAB-Map is that it can be unreliable at times and may lose track of the position for multiple reasons such as, but not limited to, lack of detected features in the observed environment, rapid movements of the camera or intense oscillations. When this occurs, a solution is to go back to where the tracking was lost. This solution is not always possible however, and even when it is possible, this is generally not an efficient and desired behavior. For this reason, a fusion of the visual odometry, the robot's internal odometry and the error between those reference frames is used to improved the overall odometry.

Since the camera is rigidly linked to the robot by a transformation T_v^c, we can write $T_v^o = T_v^c * T_c^o$, where T_v^o, T_c^o are the homogeneous transformation matrices between the map frame and, respectively, the robot frame as computed by the visual odometry and the camera frame. The previous equation can be rewritten to include the encoders-based robot odometry T_r^o, that does not take into account slipping, drift and other real world

errors :

$$\begin{aligned} T_v^o &= T_v^c * T_c^o * T_r^{o-1} * T_r^o \\ &= T_v^r * T_r^o, \end{aligned} \tag{3.14}$$

where T_v^r is the error between the encoders odometry and the visual odometry. Since this equation only holds while the visual odometry is valid, the last valid T_v^r at time $t = t_{lost}$ is used when a loss occurs at t_{lost} :

$$T_v^o(t) = \begin{cases} T_v^r(t) * T_r^o(t) & \text{if } T_v^c \text{ exists,} \\ T_v^r(t_{lost}) * T_r^o(t) & \text{otherwise.} \end{cases} \tag{3.15}$$

This approach consistently provides smooth odometry. Even when the visual information is abruptly discontinued, it continues to generate sufficiently accurate localization data until an adequate image or a reset command is processed by the SLAM library and the visual odometry is restored.

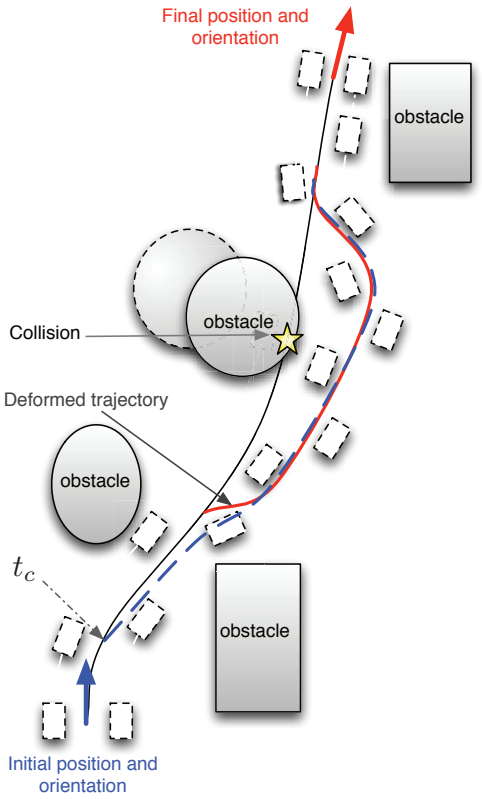


Figure 3.7 Replanning in case of collision detection : t_c is the instant at which a collision is foreseen, the new collision-free trajectory is in dashed-blue line, the deformed trajectory is in red line.

3.5.4 Dynamic Collision Avoidance

A collision might occur if an obstacle has been moved or a drift from the planned trajectory happens. To check for collision, the robot and cart footprints are projected on the ground. Then, the velocity and the direction are used to interpolate the displacement and potential collisions. When a collision is foreseen, a replanning is necessary as shown in Fig. 3.7. Eventhough, at first glance, the support polygon appears to be increased by adding the cart, the robot's support polygon is always defined by the contact between the feet and the ground. This is because the robot's arms are not fully bended, therefore the robot could fall forward or backward.

The new collision-free trajectory is found by the ARA* algorithm from the goal to the point at which the collision has been predicted. If the potential collision is due to drift, the Dijkstra grid does not need to be recalculated, therefore greatly accelerating the replanning. As the walking pattern trajectory for a humanoid robot cannot be changed instantly, a time interval T_c is required to change the planned footprints. In the implementation of ZMP preview control, a finite time horizon of two steps is used to compute the CoM trajectory. Therefore, if a collision is foreseen at instant t_c, the new collision-free trajectory provided by the ARA* algorithm is deformed to keep the next two footprints unchanged, as shown in Fig. 3.7. The robot will however stop if the deformed trajectory is in collision.

3.6 Results

Experiments were conducted on a Nao humanoid robot (Fig. 3.9), manufactured by Aldebaran Robotics [Gouaillier *et al.*, 2009]. Its dimensions are 573 mm of height, 311 mm of width and 275 mm of depth, for a total weight of 5.2 kg. The two arms as well as both legs have each 5 DOF, while the head has 2 DOF and the pelvis and hands have 1 DOF each. In addition to the above-mentioned 25 DOF, the Nao possesses two cameras, four sonars, four force sensitive resistors under each feet, two speakers and four microphones. An Inertial Measurement Unit (IMU) provides odometry data and 36 magnetic rotary encoders give joint angle information with a precision of 0.1°. On top of its head, we added an Asus Xtion Pro Live consumer-level depth camera.

The cart-like object, shown in Fig. 3.9, is a mini table 600 mm long by 300 mm wide. On one side, the two legs are 300 mm high and are set on omnidirectional 40 mm by 20 mm ball wheels. On the other side, the two legs are half the length so that the Nao robot can fully support its side of the table.

(a) Starting pose without load (b) Ending pose without load (c) Starting pose with load (d) Ending pose with load

Figure 3.8 Robot starting and ending poses for the automatic load estimation trial using turning in place motion.

Figure 3.9 The Nao robot holding the cart-like object and a load.

3.6.1 Increase in Carrying Capacity

The primary objective of our approach is to increase the maximum payload carried by a humanoid robot, without destabilizing it, while maintaining sufficient flexibility and agility. For that purpose, an experiment aimed at measuring the maximum carrying capacity of the Nao robot without any modifications has been carried out. Nao is placed in the same pose as in Fig. 3.9, then a small board (333 g) is attached to both hands and is used to support various amount of calibration weights. Even with low weight, the robot rapidly falls down or has difficulties to follow planned trajectories. At an additional 300 g weight, however, Nao falls within the first steps every run, which was determined to be its maximum carrying capacity.

With the introduction of the cart, two sets of motion primitives are available : the omnidirectional and the heavy load sets. The former generally yields shorter and faster paths than the latter, but as their names suggest, the heavy load set allows an increased maximum load capacity. The load at which the friction becomes too high to consider the omnidirec-

tional set is 700 g. Starting from a load of 700 g and up to 7,000 g, excluding the cart weight of 1370 g, the heavy set must be used since Nao is only able to move forward and rotate around the wheels pivot. In reality, the robot could push higher load, however the wood structure of our cart cannot support more than a total of 8,370 g at which its structural integrity is compromised. We are however confident that the robot could push a higher load without that constraint. To summarize, the maximum carrying capacity of the Nao robot alone is 633 g and by using the cart, it is 7,000 g, which is 11 times its normal capacity.

3.6.2 Automatic Primitive Sets Selection

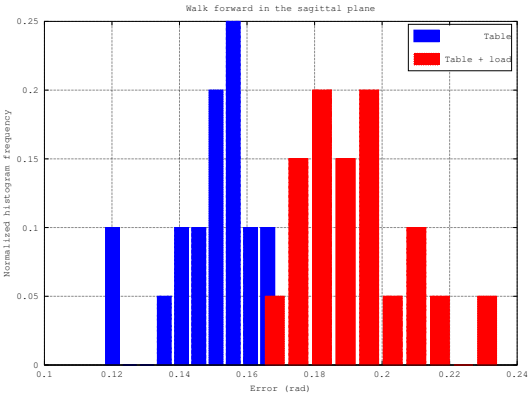
To verify our approach of sensorless automatic selection of the most appropriate primitive set, we executed a series of 20 startups without load and 20 startups with a load of 2.3 kg. Multiple movement commands have been evaluated for 10 seconds per test. These movements are : walking forward in the sagittal plane (X axis), walking in the lateral direction (Y axis) and turning in place. The latter is shown in Fig. 3.8 to illustrate the difference between the end poses with and without load.

It can be observed from Table 3.2 and Fig. 3.10 that turning in place yields the most distinctive results. By approximating the probability density of distribution functions, in the case of turning in place, by normal distributions, a threshold $x = 2.645$ rad is obtained using (3.3).

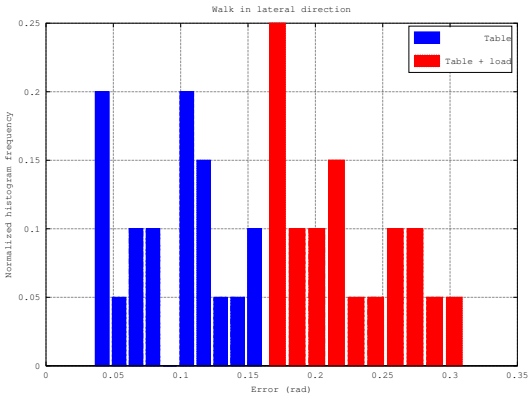
To validate the above-mentioned threshold, we conducted another series of 10 startups, and every test finished with a successful choice of the appropriate primitives set.

Tableau 3.2 Linear and angular errors, with and without a load, for all three basic motion candidates (std stands for standard deviation)

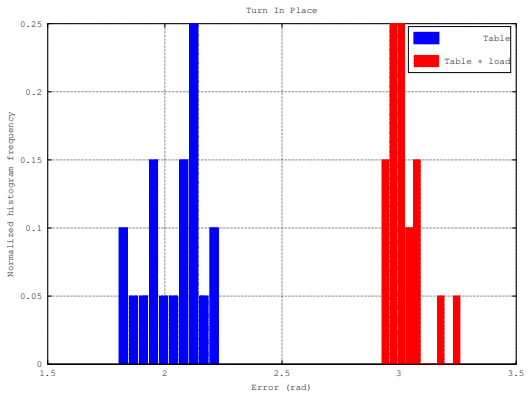
	X axis	Y axis	Rotate in place
No load			
Linear error (m)	0.151	0.133	0.156
Linear error (std)	0.011	0.022	0.040
Angular error (rad)	0.091	0.073	2.078
Angular error (std)	0.044	0.036	0.117
2.3kg load			
Linear error (m)	0.190	0.142	0.081
Linear error (std)	0.016	0.012	0.027
Angular error (rad)	0.069	0.216	3.014
Angular error (std)	0.037	0.054	0.076



(a) Walking forward in the sagittal plane



(b) Walking in the lateral direction



(c) Turning in place

Figure 3.10 Normalized histograms of the trials angular errors.

3.6.3 Articulating the Arms

In the case of omnidirectional set, the hands and arms are strong enough to articulate the cart while turning to obtain smooth trajectories. The maximum angle at which our robot

can turn the cart is 30 degrees, as illustrated in Fig. 3.11. Over that limit, one hand is colliding with the torso while the other lies outside of the robot workspace.

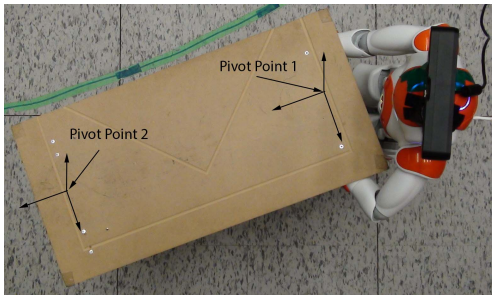


Figure 3.11 The arms posture while turning the cart at maximum angle (30 degrees).

While walking in a straight line of 1 m, the Nao robot is affected by significant linear and angular drift. The amount of drift differs depending on whether the robot is walking alone, is using the cart or if there is a load on the cart. This information is summarized in Table 3.3. Without any corrections, this error would lead the robot to constantly diverge from the planned trajectories. The odometry fusion technique explained in Section 3.5.3 is therefore used to reduce the natural drift of the robot odometry alone.

Tableau 3.3 Drift affecting a Nao robot while walking in a straight line of 1 m

	No Cart	With Cart	Cart & Load
Angular drift (°)	12.67	12.75	9.33
Linear drift (cm)	22.03	3.85	11.47

While pushing heavy load, the robot cannot however rotate in place or move laterally to cancel any drift errors that accumulate over time ; a quick replanning is therefore executed when the robot diverges too much from the planned trajectory.

As showed in Fig. 3.12(a), without any hand position correction, the lateral swing causes large oscillations that are transmitted to the table and the load. To prevent this problem, the hands are controlled to follow stable trajectories using the whole-body control scheme explained in Section 3.4.2. However, as it can be seen in Fig. 3.12(b), the error is not completely cancelled. This is mainly because : I) the hands trajectory are second priority tasks ; II) Nao has only 5 DOF in each arm ; III) the backlash of Nao motors ; IV) the time response of the motors. To minimize the impact of the backlash and the motors time response, a PID (Proportional-Integral-Derivative) controller has been implemented in the arms joint trajectory tracking controllers. Since the PID role is local by minimizing the error between the desired and the executed positions, it does not interfere with the

formulation and the performance of the whole-body control scheme. To verify the efficiency of the hands control on swing reduction, it has firstly been tested on a simulated Nao robot.

Without any correction, the average peak-to-peak position movement of the hands is 47.6 mm. However, with the whole body control, the hand distance from desired position has been reduced to 16.4 mm, reducing the hand oscillations by 65.5%. By adding the PID controller with coefficients $Kp = 2$, $Ki = 0.01$ and $Kd =$ -0.015, the error has been reduced further to 76.1% for an average peak-to-peak movement of 11.4 mm. As a result, significantly less oscillations are transmitted to the table, leading to a safer and enhanced carrying ability and load stability.

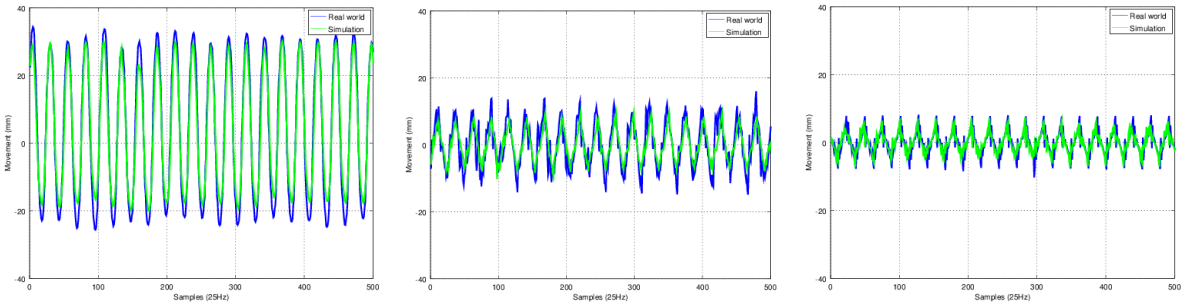
The same tests were then conducted on the real robot. The results of the simulated and real world test are summarized in Table 3.4. It can be seen that the results are quite similar, thus the proposed technique significantly increases the load stability.

Tableau 3.4 Simulated and real world hands corrections improvements

	Without Corrections	No PID	With PID
Simulated			
Peak-to-peak movement (mm)	47.6	16.4	11.4
Improvement (%)	0	65.5	76.1
Real world			
Peak-to-peak movement (mm)	56.0	23.3	15.6
Improvement (%)	0	58.5	72.1

3.6.4 Navigating in a Cluttered Environment

To test the system as a whole and to validate the proposed algorithms, we conducted three series of five experiments. In each experiment, the robot starting and goal positions were chosen in a way that the robot had to navigate through a field of objects on the ground. Each of them served as an obstacle that must be avoided by the robot and the cart. They were placed to form various feasible paths and force tight turns to take advantage of the additional degree of freedom (the rotation of the cart θ_{ca}). The three series were composed of the same experiments containing the same initial configuration of the obstacles in the environment and using the same initial and goal positions and orientations, but with different transported objects. The omnidirectional set of primitive has 12 different primitives with θ sampling of 11.25°, while the heavy set has only five possible primitives, but with a precision of 5.625°.



(a) No hands correction (b) Position control correction (c) Position control and PID correction

Figure 3.12 Influence of hand corrections on table oscillations.

Figure 3.13 shows the start and end positions for each type of experiment. In this figure, the obstacles are in black, while the red areas around them are inflation zones where the cost is higher than free space to prevent the robot from passing too close to obstacles. These zones are used as a security buffer and the center of the robot and the cart should avoid, if possible, plan to pass inside it. The cyan zone is a lethal zone, because if the center of the cart or the robot enters it, it means that an edge is in collision with an obstacle.

The first series consists of the Nao robot alone, without a cart or load. It chose the omnidirectional primitives set to navigate through the obstacles. The second one was conducted with the Nao holding the cart, which significantly increased the navigation footprints. These tests also chose the omnidirectional set to build the plan. For the third and final series, the robot was pushing the cart with an additional load of 2,300 g. In this case, the heavy load set of primitives have been chosen to allow the robot to properly plan a trajectory in the cluttered environment.

The ARA* planner initial $\epsilon = 3$ means that the suboptimal solution cannot be worse than 3 times the cost of the optimal solution. A time limit of 10 seconds was chosen and within that time, the planner converged to the optimal solution at every run (ϵ successfully decreased to 1). For each generated path, we measured the total time to execute the trajectory, the trajectory length, the initial solution time, the optimal solution time, the initial expanded nodes and the final expanded nodes. These results are summarized in Table 3.5.

Even though all trajectory lengths are very similar, the time needed to complete the trajectory is greater with the cart than without it, and even greater with a load. This is explained by the friction on the cart wheels causing slippage as the robot tries to move and

Tableau 3.5 Experimental statistics

	No Cart	With Cart	Cart & Load
Total time (s)	60.72	86.01	135.9
Total time std (s)	4.64	9.85	18.15
Trajectory length (m)	2.29	2.24	2.96
Trajectory length std (m)	0.09	0.28	0.27
Average velocity (m/s)	0.0377	0.0260	0.0217
Initial solution time ($\epsilon = 3$) (ms)	2	6	18
Initial solution time std (ms)	4	9	30
Optimal solution time ($\epsilon = 1$) (ms)	166	172	200
Optimal solution time std (ms)	165	192	170
Initial node expansions	57.0	218.0	1782.6
Initial node expansions std	49.7	350.3	2094.1
Total node expansions	5660	6350	16717
Total node expansions std	7426	8539	16396

slowing its movements down. Every step in a direction results in a slippage in the opposite direction, thus progressing less distance with each step. This also holds true while rotating in place. Increasing the load weight amplifies this effect of slippage, slowing the Nao down even more. This leads to a reduced speed of 31% comparing to the empty cart and 42% speed reduction with the additional 2.3 kg load.

Although the primitives with the robot alone and with the cart are the same, it is hard to find a path as optimal while using the cart than without it. Some places might not be accessible or Nao might need more time to clear obstacles before turning because the cart is in the way. However, since the average length is about the same though, moving with the table does not impair much of the robot ability to travel the cluttered environment swiftly.

The weight primitives trajectory length is, however, higher in comparison, about 29% higher than with the Nao alone. This is primarily due to the change of primitives. The movement is not as smooth, and in particular, moving sideways or in diagonal become impossible. Also, turning in place versus turning around a pivot placed $r = 0.60$m away increased the trajectory by at least $L = \frac{\theta \pi r}{180}$ every time the robot makes a turn.

Even though in our case the trajectories are quite small, the use of sub-optimal solutions has proven to be significantly faster. Indeed, for our experimental settings, it takes about 83, 28.7 and 11.1 times less states expansions and is 99.3, 29.1 and 9.4 times faster to compute the first solution for $\epsilon = 3$, as opposed to the optimal solution for the robot alone with an empty cart and with a load, respectively. This could be especially useful for real life large scale distances and experiments where quick reactions and planning are necessary.

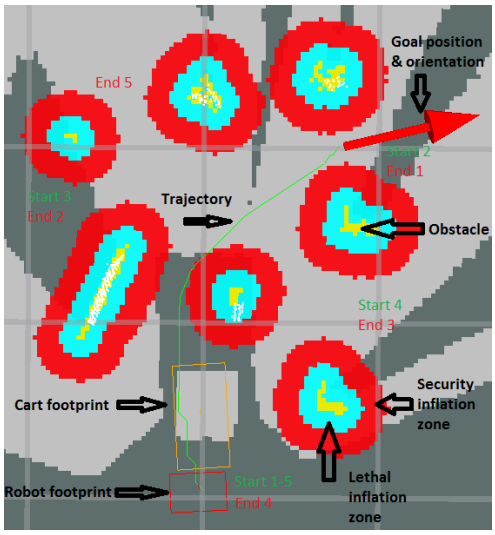


Figure 3.13 Map of the starting and ending positions with obstacles, lethal and security inflation zones around them, the Nao and cart footprints and goal position/orientation.

The output from the SLAM library are shown in Fig. 3.14 and Fig. 3.15. The localization information is displayed as a continuous red line and the mapping information is represented by the rest of the point cloud built incrementally. It can be observed that mapping, while pushing the table, obstructs the ground and thus yields a less practical and complete map of the environment. However, since it is possible to use the side of the table for localization, thanks to our filtering technique, the localization data remained continuous and reliable for all the tests. It is important to note that the ground in the testing room had sufficient texture and patterns to produce reliable data using the SLAM library. When tested on a plain ground, RTAB-Map could not, however, extract enough features to use visual localization with only the obstacles.

Also, the friction on the wheels is not high enough to completely prevent them from sliding and thus to act as a perfect pivot while pushing a heavy load. This caused additional errors when turning and walking, and forcing a replanning when the accumulated error became significant. For this reason, with the heavy load, three replanning were needed on average, while no replanning was necessary with or without the cart when no load was carried.

When an obstacle is moved, a replanning is essential to avoid collision. Figure 3.16, Fig. 3.17 and Fig. 3.18 show snapshots of the Nao navigating with the different possible configurations, i.e., without cart, with the cart only and with the cart supporting a heavy load, while successfully avoiding moving obstacles. In these figures, a new path is recomputed

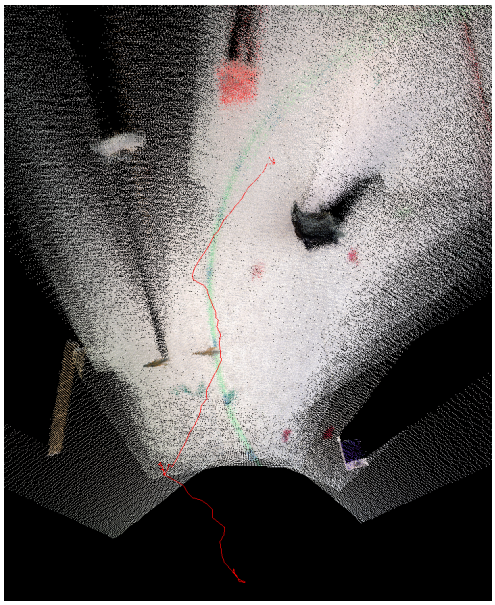


Figure 3.14 Localization (red line) and mapping (point cloud) for the robot alone.

before any collision occurs and the Nao continues its way around. In all our experiments, no collision between neither the Nao nor the cart with any obstacles occurred.

3.7 Conclusion and Future Work

In this paper, we present a system capable of carrying a heavy load on an articulated cart while navigating in a cluttered environment. We also gave practical insights into the implementation of the proposed approach on a real humanoid robot. Our method uses two different sets of primitives to plan a trajectory : a low to medium weight omnidirectional set and a more restrictive heavy weight set. We also presente a sensorless technique to test a difficult movement on startup and accurately determine the appropriate set of primitives to be used. It is based on the error between the desired and the executed commands.

When moving throughout the environment, a depth camera and a SLAM library, map the obstacles in real-time and provides a visual odometry. This information is then fused with the robot odometry to provide a consistent, continuous and reliable odometry data. While mapping the environment, the cart pushed by the robot is filtered and ignored by the library to prevent it from being considered as a permanent obstacle. It would be possible to integrate the SLAM, the 3D occupancy grid and the loop closure algorithms provided by RTAB-Map with the memory efficient OctoMap [Hornung *et al.*, 2013] for planning

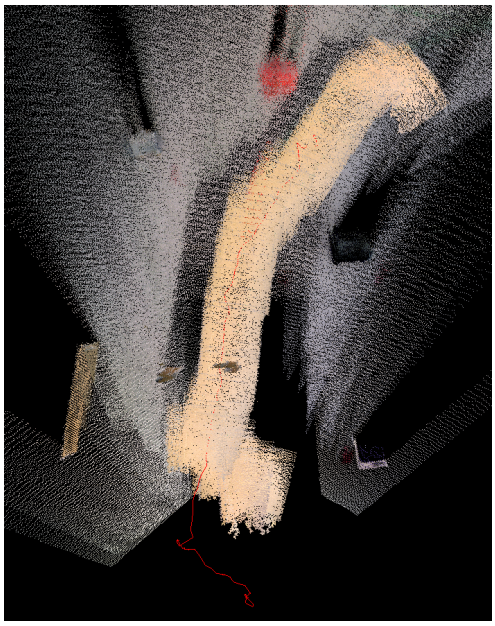


Figure 3.15 Localization (red line) and mapping (point cloud) for the robot with the table.

and obstacle detection. However, since the project is mainly in 2D for the moment, this integration was not performed.

Moreover, by controlling the hands adequately using a whole-body control scheme coupled with a PID, we were able to articulate the cart in tight turns and to significantly reduce the lateral swing from propagating to the load.

In future works, to make the robot completely autonomous, it is necessary to implement it entirely on the Nao's internal CPU. The biggest challenge is the limited processing power of the platform in comparison to other high-end humanoid robots that often have multiple onboard computers. The possible solutions are either to change the computing hardware to increase its processing power, or to modify the algorithms to be computationally efficient and optimized for the current platform. The most demanding part of the system for the CPU, without a doubt, is the processing of the vision based odometry. Using different parameters in the library to decrease the precision and frequency of the mapping and localization would lead to improve the performance regarding the computing power, but would necessarily have a negative impact on the results. Moreover, integrating a hierarchical quadratic programming scheme [Escande *et al.*, 2014] to solve the whole-body control problem will be also investigated.

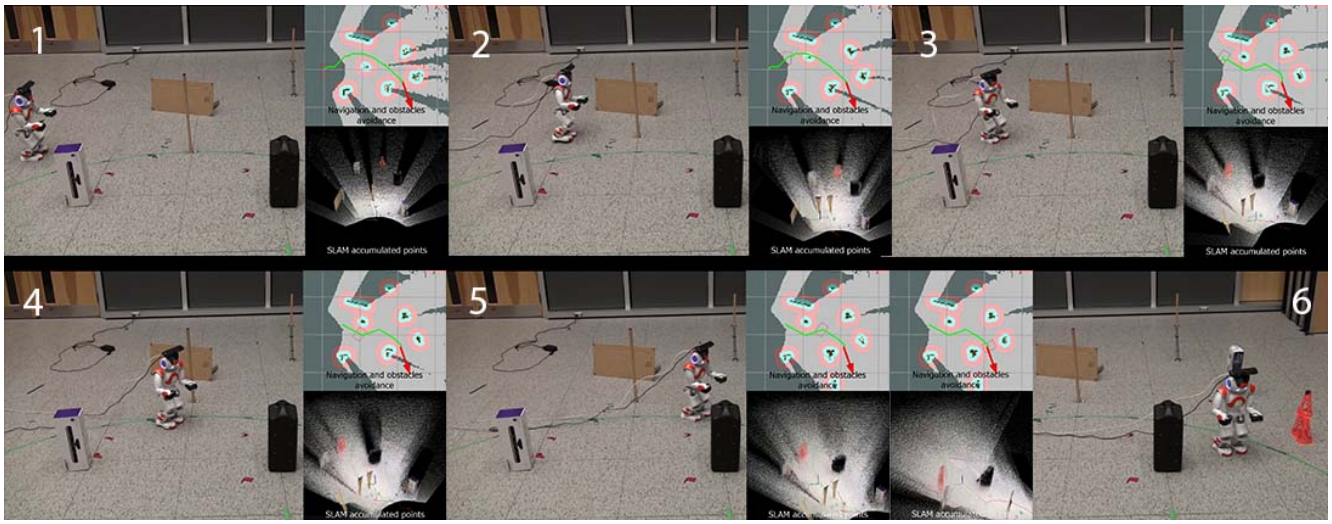


Figure 3.16 Snapshots of the Nao navigating without the cart using omnidirectional primitives.

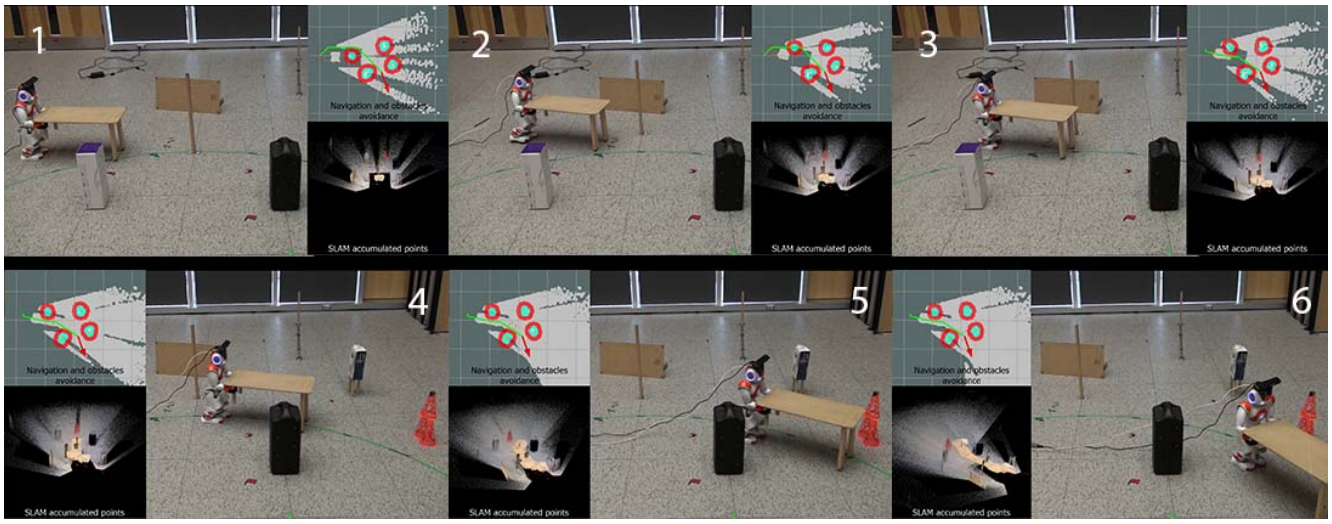


Figure 3.17 Snapshots of the Nao navigating with the cart using omnidirectional primitives.

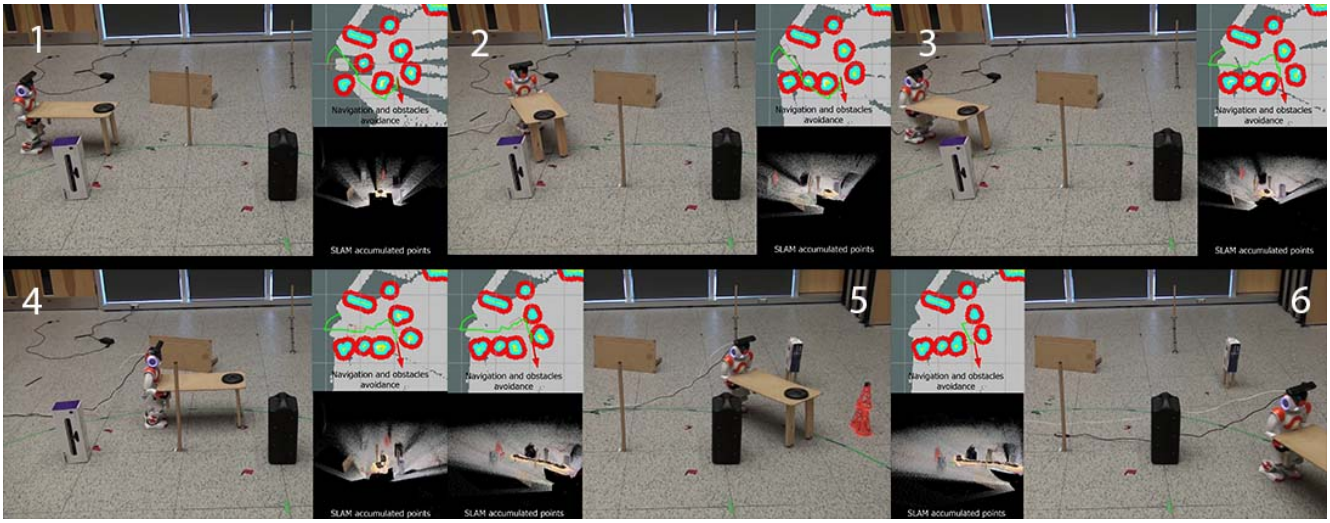


Figure 3.18 Snapshots of the Nao navigating with the cart using heavy load primitives.

CHAPITRE 4

Transport coopératif d'un objet par deux robots humanoïdes dans un environnement encombré

Auteurs et affiliation :

A. Rioux : étudiant à la maîtrise en génie électrique, Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.

W. Suleiman : professeur, Université de Sherbrooke, Faculté de génie, Département de génie électrique et de génie informatique.

C. Esteves : professeur, Universidad de Guanajuato, Département de Mathematiques. Mexique

J-B. Hayet : professeur, Centro de Investigación en Matemáticas (CIMAT), Guanajuato, Mexique

Date de soumission : 18 février 2016

Revue : IEEE/ASME Transactions on Mechatronics

Titre anglais : Cooperative SLAM-Based Object Transportation by Two Humanoid Robots in a Cluttered Environment

Contribution au document : Nos contributions sont : (1) un algorithme de planification de mouvement multi-robots de faible dimension afin de trouver une trajectoire sans collision en utilisant la carte construite de l'environnement ; (2) un algorithme pour produire des données continues et homogènes d'odométrie, en fusionnant les informations visuelles et l'odométrie du robot ; (3) un système de synchronisation qui utilise le décalage relatif des robots, la projection des robots sur la base de leur position des mains ainsi que l'erreur de rétroaction visuelle calculée à partir d'une caméra frontale ; (4) un contrôle en temps réel efficace de tout le corps pour contrôler les mouvements du système robot-objet-robot en boucle fermée.

Résumé français : Bien que ces dernières années il y a eu quelques études destinées à la navigation de robots dans des environnements encombrés, seulement quelques-unes d'entre elles ont abordé le problème de robots qui naviguent en déplaçant un objet volumineux ou lourd. Ceci est particulièrement utile pour le transport d'objets de différentes formes et poids sans avoir à modifier physiquement le robot.

Dans ce travail, nous abordons le problème de faire naviguer deux robots humanoïdes dans un environnement encombré tout en transportant un très grand objet qui ne peut tout simplement pas être déplacé par un seul robot. Nous présentons un système de navigation complet, de la construction progressive d'une carte de l'environnement et de calcul des trajectoires sans collision, à la conception de la commande pour exécuter ces trajectoires. Nous présentons des expériences faites sur des robots Nao, équipés de capteurs RGB-D montés sur leurs têtes, se déplaçant avec un objet tout en contournant les obstacles. Nos expériences montrent qu'un objet de taille non négligeable peut être transporté sans physiquement changer le robot, améliorant donc la capacité des robots humanoïdes dans des situations réelles.

4.1 Abstract

Although in recent years there have been quite a few studies aimed at the navigation of robots in cluttered environments, few of these have addressed the problem of robots navigating while moving a large or heavy object. This is especially useful when transporting objects of different shapes and weights without having to change the robot's hardware.

In this work, we tackle the problem of making two humanoid robots navigate in a cluttered environment while transporting a very large object that simply could not be moved by a single robot. We present a complete navigation scheme, from the incremental construction of a map of the environment and the computation of collision-free trajectories to the design of the control to execute those trajectories. We present experiments made on Nao robots, equipped with RGB-D sensors mounted on their heads, moving an object around obstacles. Our experiments show that a significantly large object can be transported without changing the robot's main hardware, and therefore enhancing the humanoid robot's capacity in real-life situations.

Our contributions are : (1) a low-dimension multi-robot motion planning algorithm to find an obstacle-free trajectory using the constructed map of the environment as an input ; (2) a framework to produce continuous and consistent odometry data, by fusing the visual and the robot odometry information ; (3) a synchronization system that uses the projection of the robots based on their hands position coupled with the visual feedback error computed from a frontal camera ; (4) an efficient real-time whole-body control scheme to control the motions of the closed-loop robot-object-robot system.

4.2 Introduction

One of the advantages of having arms on a robot is that it can manipulate a load. This capacity can be useful for a wide range of actions, including transporting objects from one place to another. However, using one robot only, the maximum payload is generally low and the size of the transported objects is necessarily limited. One way to deal with this issue is to distribute the weight or surface area held to multiple robots. In this work, we deal with the more specific problem of having two humanoid robots cooperating to maneuver a bulky object through obstacles.

Many studies have been done on robots moving objects to a specific goal. However, most of them are executed using multiple wheeled robots that position themselves around the object to push it in the desired direction [Kube et Bonabeau, 2000; Miyata *et al.*, 1997;

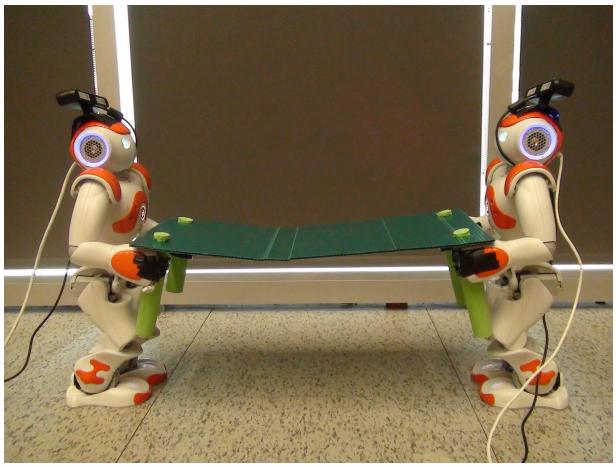


Figure 4.1 The Nao robots holding an object together

Ota *et al.*, 1995; Wang *et al.*, 1999; Yamashita *et al.*, 2003]. In the aforementioned works, the manipulator comes in contact with the object at only one point, which barely allows any control over the object while moving and manipulating it. Holonomic wheeled robots are much less complex to control than humanoid robots and are mainly used here to slide box-like objects on the ground. Robots with humanoid arms have a better control on the structure of an object and are, therefore, may be more suitable to control objects within a large range of different shapes.

More advanced and specialized models of wheeled robots can possess humanoid torsos and arms which give them the same capacity to keep a high level of control when holding or transporting objects [Hirata et Kosuge, 2000] [Suda et Kosuge, 2002] [Lawitzky *et al.*, 2010]. However, most environments made by and for humans are more suitable for fully humanoid robots. To this end, the subject of transporting bulky or heavy objects with a humanoid robot has received attention in the past years. To achieve this task, many different techniques have been developed. For instance, lifting the load from the ground [Harada *et al.*, 2005] [Ohmura et Kuniyoshi, 2007] [Yoshida *et al.*, 2008] or using a part of the transported object as a pivot to move it [Aiyama *et al.*, 1993] [Yoshida *et al.*, 2006] [Yoshida *et al.*, 2010].

Other works have explored the use of humanoid robots working in cooperation to transport an object. One of the most popular control schemes for cooperation with multiple robots, big or small, is the leader-follower control scheme. One of the robots, the leader, based on its position and its surrounding, computes the common plan of the system or is being directly controlled by a human operator. The second robot, the follower, simply follows the leader robot. While relatively simple to implement, this technique has many flaws when used in closed-loop cooperation, which is why other works have used disjoint objects [Inoue

et al., 2007] or constrain the robots motions to one axis [Wu *et al.*, 2011]. Since the follower only responds to the leader's movement after it has already started, a significant time delay is introduced. Moreover, interpretation errors of the leader's movements can destabilize the closed-loop system and cause unexpected falls.

Another possible control scheme for this problem is the set of synchronized controllers [McGill et Lee, 2011] [Wu *et al.*, 2014]. With this technique, there is no apparent master robot, and an external centralized controller manages all the robots simultaneously using on the complete information provided by the robots. As a result, the synchronized robots start, move and stop together. This way, the system is highly responsive and any errors are shared within the whole system. To achieve this, one can model the entire robot-object-robot system as a quadruped with a rigid body, thus preventing the robots to manipulate the object freely [McGill et Lee, 2011]. The dynamic of the system is simplified, and many degrees of freedom can be removed by adding virtual constraints. The work of [Wu *et al.*, 2014] explored this strategy with the human-sized HRP2 robot. Even though their results look promising in simulation, their implementation does not consider navigation among obstacles nor the robot localization and mapping. It also makes an extensive use of expensive six-axis force sensors located in the wrist, which makes the approach difficult to generalize to many affordable robots that are not equipped with such force sensors.

The main contribution of our work is to provide a complete framework for cooperative autonomous humanoid robots navigation in a cluttered environment, while manipulating an object in a closed kinematic chain. In addition, an improved odometry data fusion scheme in presence of unreliable real-time SLAM information is detailed. Finally, a synchronization mechanism, which uses the arms, trajectories, robots reflection positions and relative visual positions is also proposed.

To address the problem of making a closed-loop robot-object-robot navigate in a cluttered environment, the following sub-problems need be solved : (I) planning the trajectories ; (II) controlling the object ; (III) sensing the environment ; and (IV) synchronizing the system. This paper is organized according to these sub-problems. Section 4.3 presents an anytime search-based planner that exploits a given set of motion primitives. This planner considers both robots and the object footprint to plan a safe trajectory between obstacles. In Section 4.4, we show how real-time information from a entry level depth camera is used to perform simultaneous localization and mapping (SLAM) of the surrounding obstacles in the cluttered environment. Section 4.5 details the different synchronization mechanisms and the way they are integrated together. Section 4.6 describes how to determine the humanoid robots' footprints, and then how to compute the robots' feet and hands trajec-

tories to minimize the swing effect, synchronize the motions of the two robots, and follow the planned trajectory by using a task priority whole-body control scheme. Finally, in Section 4.7, results of simulation and real world experiments are presented and discussed.

4.3 Planning a Valid Path

To navigate through a cluttered environment, the computation of a collision-free path for both robots is essential. For this, we chose a lattice-based graph planning using an ARA* search [Likhachev *et al.*, 2004], motivated by the use of motion primitives ensuring feasible robot-object-robot configurations and transitions. The environment is modelled by a 2D grid cost-map that discriminates obstacles from free space by using a fixed threshold on the cost value. Moreover, it allows obstacle inflation to increase the security margin.

4.3.1 State Representation

Each node of the search graph needs a representation of a full state including the configurations of both robots and of the object in the plane. To achieve this, it is possible to model the state in $\mathbb{R}^2 \times \mathbb{S}^1 \times \mathbb{R}^2 \times \mathbb{S}^1 \times \mathbb{R}^2 \times \mathbb{S}^1$:

$$s = (x_{r1}, y_{r1}, \theta_{r1}, x_{ob}, y_{ob}, \theta_{ob}, x_{r2}, y_{r2}, \theta_{r2}), \tag{4.1}$$

where x_{ri}, y_{ri} and θ_{ri} $(i = 1, 2)$ are the positions and orientation of the i-th robot, and x_{ob}, y_{ob} and θ_{ob} are those of the object. Note that these simplified, planar models for the robots and the object are used for the planning step only.

As the working space of our robot's arms is too small to fully take advantage of both rotation and translation, the system is simplified by setting a pivot point at the middle of the pair of hands for both robots, as shown in Fig. 4.2. The closed-loop grasping of the robot on the table is shown in Fig. 4.1. The pivot points positions have been chosen to maximize the rotation range within the robot workspace, resulting in a smaller state space $\mathbb{R}^2 \times \mathbb{S}^1 \times \mathbb{S}^1 \times \mathbb{S}^1$:

$$s = (x_{r1}, y_{r1}, \theta_{r1}, \theta_{ob}, \theta_{r2}). \tag{4.2}$$

Even though the above simplification removes the ability of the object to translate on the plane independently, the robot retains enough manipulability to minimize the robot-object-robot collision area around obstacles. The simplified state representation of Equation (4.2) is shown in Fig. 4.3.

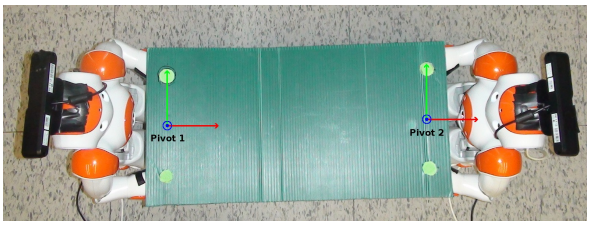


Figure 4.2 Top view : Pivots position.

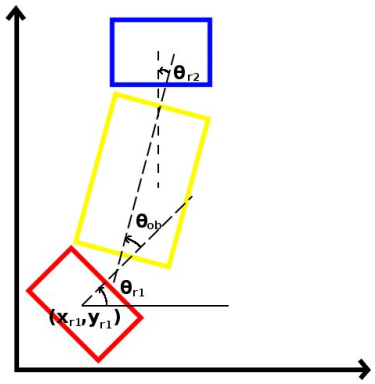


Figure 4.3 Simplified state representation. In red and blue : the two humanoid robots. In yellow : the load.

4.3.2 Transition Model

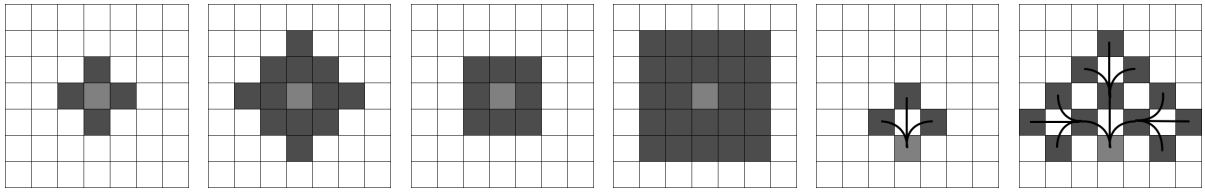


Figure 4.4 Different forms of graph search methods. From left to right : Von Neumann neighborhood 1st & 2nd expansions ; Moore neighborhood 1st & 2nd expansions ; example of a set of motion primitives 1st & 2nd expansions.

In a lattice-based graph planner, transitions between nodes are triggered by actions chosen within a finite fixed-set of motion primitives. Motion primitives allow the decomposition of complex motion generation in robotics and it is very likely that they are also used by humans and animals [Hart et Giszter, 2010]. An important feature of methods based on the lattice representation is that all connections are feasible paths. The way the expansion of nodes is done during the search encodes the aforementioned notion of motion primitives. An example of the first two expansions of different graph search methods is shown in Fig. 4.4. The representation we chose (illustrated by the two expansion images on the right) is really suitable for highly constrained systems, such as a system of two robots transporting an object, in contrast to other commonly used forms of encoding transitions in graph

search, including Von Neumann neighborhood [Weisstein, 2012] or Moore neighborhood [Weisstein, 2005] (illustrated by the expansion images on the left).

The full set of motion primitives used for this problem is showed in Fig. 4.5. It includes (a) forward, backward, sideway motion and every diagonal motion, (b) rotations around each robot and around the object center, and special movements such as (c) C-turns and (d) S-turns. The last two are more complex and make use of the hand articulations to increase agility around obstacles. Executing complex motions specific to a system in a coherent and logical way is the main reason why we use motion primitives, over a more traditional method doing a homogeneous sampling of the system DOFs. The joints of the system that allow these particular motions are the aforementioned pivot points.

4.3.3 Path Cost Function

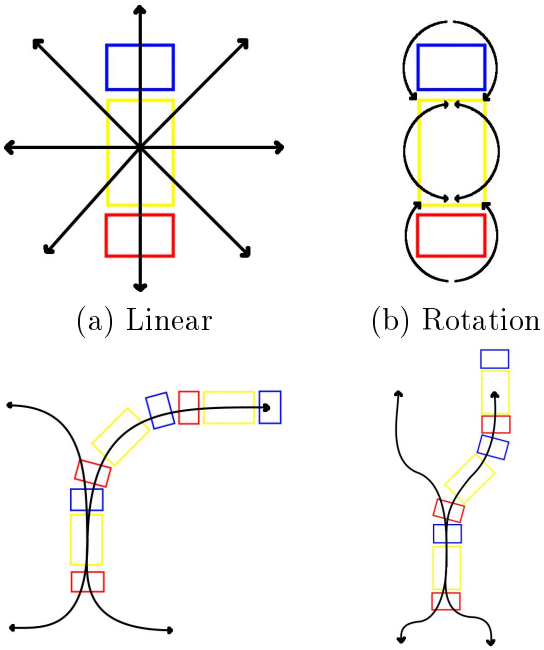
The cost of a transition from state s to s' is based on the time to execute that transition and is computed as follows :

$$g(s,s') = \begin{cases} \frac{\sqrt{(\Delta x_{r1})^2+(\Delta y_{r1})^2}}{\dot{\boldsymbol{r}}_{\boldsymbol{1}}{}^{+}} \times DF & \text{if } \Delta x_{r1} \neq 0 \text{ or } \Delta y_{r1} \neq 0 \\ \frac{\sqrt{(\Delta x_{r2})^2+(\Delta y_{r2})^2}}{\dot{\boldsymbol{r}}_{\boldsymbol{2}}{}^{+}} \times DF & \text{otherwise} \end{cases} \tag{4.3}$$

where Δx_{ri}, Δy_{ri} are the variations of the x and y coordinates of the i-th robot pelvis, between states s and s', DF is a difficulty factor associated with each primitive, and $\dot{\boldsymbol{r}}_{\boldsymbol{i}}{}^{+}$ is the robot maximal linear velocity. This ratio gives us the approximate time to execute the primitive. To avoid collision with obstacles and keeping a safety distance to those obstacles, transitions close to obstacles have higher costs. Note that only one primitive is selected to connect two states s and s'.

Since the state representation does not contain any information about the position of the second robot, we must determine Δx_{r2} and Δy_{r2}, if needed, with the first robot position. To do so, we compute the projection of the object by rotating the first robot's pelvis frame by θ_{r1}, given a hand frame rotation of θ_{ob}. Then, we project the second robot around the object given a hand frame rotation of θ_{r2}.

The difficulty factor DF is a special value set by the system's expert in order to prioritize or penalize certain motions, which results in a smoother and a more natural looking trajectory. For instance, turning in place then moving forward (Fig. 4.6 a) takes a longer time than moving in diagonal (Fig. 4.6 b). However, over a long distance, the former reduces the trajectory footprint and is therefore more natural looking while reducing the



(a) Linear (b) Rotation

(c) C-turn (d) S-turn

Figure 4.5 Set of possible motion primitives.

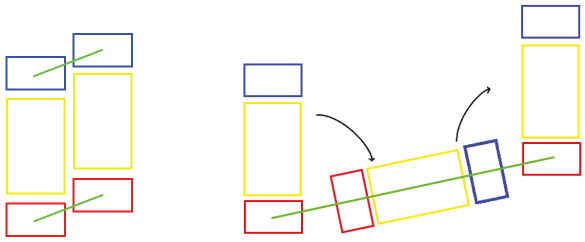
chances of drifts caused by the table movements. For those reasons, moving sideway has a higher DF than turning and moving forward. In sum, for small distances, the time cost takes over the DF; however, for long distances, it is more likely that turning in place and moving forward would be preferred. Examples of DF values are given in Table 4.1.

Tableau 4.1 The DF factor for different motion classes

Primitive set	DF factor
Forward	1
Turn in place	2
Backward	3
Sideways	2
Diagonal	1

4.3.4 Search Algorithm

A* is one of the most popular search algorithms. In addition to the use of a path cost function, a heuristic biases the search towards the most promising states. Even though A* is optimal when it finds a solution, that solution does not always exist or may not be found within a reasonable time. The Anytime Repairing A* (ARA*) focuses on delivering a suboptimal solution as fast as possible. This solution is then optimized iteratively within a predefined limited time. Also, the states are expanded from goal to start so that the



(a) Short diagonal (b) Long diagonal

Figure 4.6 Examples of the effect of the DF.

heuristic costs remain valid when replanning and do not need to be recomputed. The cost function takes the form of :

$$f(s,s') = g(s,s') * \max_{\sigma \in \pi(s,s')} (Cost_{cell}(\sigma)) + \epsilon h(s'), \epsilon \geq 1 \tag{4.4}$$

where $\pi(s,s')$ is the path connecting s and s', $g(s,s')$ is the path cost of Equation (4.3), $h(s')$ is the heuristic that uses a 2D grid containing all the Dijkstra distance costs from the goal to the start states and

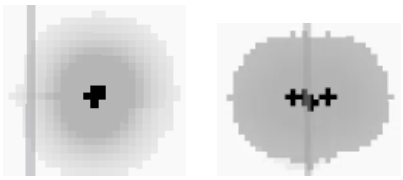
$$Cost_{cell}(\sigma) = \begin{cases} 1 & \text{free space} \\ 2 \text{ to } 99 & \text{inflation} \\ \infty & \text{obstacles} \end{cases} \tag{4.5}$$

which is the cost of cell σ that belongs to the path connecting s and s'. The search is biased towards states closer to goal (because of the term $h(s')$) and returns a solution that is, at worst, ϵ times the cost of the optimal solution.

The inflation is a zone around obstacles where all the cells have a higher cost. It is used as a security margin to bias the search farther from obstacles and reduce danger of collisions. It can be set as a decreasing gradient from the obstacles (Fig. 4.7.a) or as a fixed value in the range above (Fig. 4.7.b).

4.4 Simultaneous Localization and Mapping (SLAM)

To move in a cluttered environment, a robust and precise sensing input is primordial to determine the position of the obstacles, to detect collisions and to plan valid long term and short term paths. Also, odometry drift must be constantly verified and corrected by an accurate localization mechanism to ensure that the planned path is closely followed.



(a) Gradient (b) Solid

Figure 4.7 Types of inflation.

The human-sized humanoid robots, such as HRP-2 or the humanoids robots participating in the DARPA Challenge [Defense Advanced Research Projects Agency, 2016], are able to build a 3D map on the fly using their very sophisticated proprioceptive and exteroceptive sensors. However, the Nao robot has natively only two cameras in its head for sensing and localization. A first approach would be to use those cameras. However, this has proven to be a very difficult task [Oßwald *et al.*, 2010; Stasse *et al.*, 2006]. Indeed, a humanoid robot swings laterally while walking. This effect coupled with low resolution cameras leads to pictures of poor quality. Furthermore, the field of view of the Nao is greatly obstructed by the large object being transported and by the other robot situated right in front of it. As a result, it is hard to determine precisely the position of the environment features and of the obstacles with respect to the robot only with the Nao's cameras.

4.4.1 Real-Time Appearance-Based Mapping (RTAB-Map)

We chose to add an RGB-D camera on the top of each Nao's head and to use it for mapping [Maier *et al.*, 2012], and use the open source library RTAB-Map [Labbe et Michaud, 2014a,b]. RTAB-Map is a RGB-D Graph-SLAM library that uses a bag-of-words technique for loop closure detection. A memory management system limits the quantity of information in memory to ensure the constant satisfaction of large environment real-time constraints. With these features, it supports large maps with kilometers-long paths and multi-sessions mapping and localization.

RTAB-Map also provides a robust odometry system based on visual information. It can create 3D maps of the environment as well as 2D occupancy grids by projecting the obstacles on the ground plane. In this work, we fuse the occupancy grids generated by both robots into a common one that is used for planning. Even though our planning algorithms could use probabilistic grids, the SLAM library that we use only provides deterministic occupancy grids at this moment.

4.4.2 Odometry Fusion

A problem that may occur with the visual odometry produced by RTAB-Map is that it may lose track of the position for multiple reasons, such as missing image features in the observed environment, rapid movements of the camera or intense oscillations. When this occurs, we could go back to where the tracking was lost, but this is not efficient and may even be impossible. Hence, a fusion of the visual odometry, the robot's internal odometry and the error between those two reference frames is used to improve the overall odometry.

Since the camera is rigidly linked to the robot by a transformation T_v^c, we can write $T_v^o = T_v^c * T_c^o$, where T_v^o, T_c^o are the homogeneous transformation matrices between the map frame (index o) and, respectively, the robot frame (index v) and the camera frame (index c). The previous equation can be rewritten to include the encoders-based robot odometry T_r^o (indice r), that does not take into account slipping, drift and other real world errors :

$$\begin{aligned} T_v^o &= T_v^c * T_c^o * T_r^{o-1} * T_r^o \\ &= T_v^r * T_r^o, \end{aligned} \tag{4.6}$$

where T_v^r is the error between the encoders odometry and the visual odometry. Since this equation only holds while the visual odometry is valid, the last valid T_v^r, at time $t = t_{lost}$, is used when a loss of the visual odometry occurs at $t = t_{lost}$:

$$T_v^o(t) = \begin{cases} T_v^c(t) * T_c^r(t) * T_r^o(t) & \text{if } T_v^c \text{ exists,} \\ T_v^r(t_{lost}) * T_r^o(t) & \text{otherwise.} \end{cases} \tag{4.7}$$

This approach consistently provides smooth odometry. Even when the visual information is abruptly discontinued, it continues to generate sufficiently accurate localization data until an adequate image or a reset command is processed by RTAB-Map and the visual odometry is restored.

4.5 Synchronization

In the section, we present the different techniques applied in order to minimize the position error of the synchronized trajectories and allowed us to do successful humanoid-humanoid task cooperation. These four techniques are : object and hands stabilization ; trajectories synchronization ; synchronized reflections ; and visual feedback.

4.5.1 Object Stability and Hand Stabilization

At low speed, a humanoid robot center of mass (CoM) moves horizontally from one support foot to the other to stay in balance [Kajita *et al.*, 2003]. This lateral motion causes the entire upper body to oscillate laterally at an amplitude proportional to the distance between the center of its feet, which, in our case, causes the transported object to move by the same amplitude.

Since the object to carry is fully controlled by the robots' hands, it is possible to reduce the oscillating effect to improve the closed-loop kinematic chain stability. On the one hand, if both robots swing at the same time, synchronization is maintained easily, but the object and anything on it would swing dangerously. On the other hand, if the robots swing in any other way, the force generated by each robot movement will be transmitted to the other robot and may cause instability.

Our solution to compensate this instability without changing the walking gait is to use the robots' hands and keep them at a fixed position in space, relatively to the planned trajectory. This position is determined from the starting position of the robot and corresponds to the initial transformation between feet and hands. The output of our corresponding hand stabilization technique is $\dot{\boldsymbol{r}}_{lh}$, $\dot{\boldsymbol{r}}_{rh}$, which are the linear and angular velocities of the left and right hands, respectively. Those outputs are integrated into the whole-body control scheme described in Section 4.6.

4.5.2 Synchronization of trajectories

Even if both robots are independent and receive their own trajectory to follow, these trajectories are lined up in such a way that they are made of configurations located at a constant distance from each other. However, real robots being imperfect, they do not necessarily move at the exact same speed given the same command. For this reason, the trajectories are segmented into corresponding waypoints, each waypoint being a state of the graph plan. To progress to the next waypoint, both robots have to agree that they are close enough to their current waypoint, in terms of both position and orientation. If only one robot is near its waypoint, it slows down while converging to it, until the other robot agrees that they can proceed to the next waypoint. The output of trajectory synchronization is the vector $\dot{\boldsymbol{r}}_p$, which is the linear and angular velocity of the robot's pelvis joint.

4.5.3 Synchronized Reflections

Now that the whole system is more stable with regard to the individual swing added by both robots, the position of each robot needs to be synchronized with the other one along the planned trajectory. To do so, the reflection of each robot with respect to the other is computed by using the robots respective hands, world frames and transported object properties. First, the position of the center of the object with respect to a robot i can be found by :

$$T_{ob}^{r_i} = midpoint(T_{h_r}^{r_i} * T_{ob}^{h_r}, T_{h_l}^{r_i} * T_{ob}^{h_l}) \tag{4.8}$$

where $T_{ob}^{r_i}$, $T_{h_{\{r,l\}}}^{r_i}$, $T_{ob}^{h_{\{r,l\}}}$ are respectively the current transformations between the robot i and the object frames, the robot i and its own hands frames and the object center and the robot i hands frames. The robot hand frame may be defined for each hand (hence the notation $T_{h_{\{r,l\}}}^{r_i}$). The function *midpoint* computes the midpoint of two transforms.

With these transformations defined, we set the reflection synchronization position for each robot as follows :

$$T_{p_i}^{o} = T_{r_j}^{o} * T_{ob}^{r_j} * T_{ob}^{r_i\,-1} \qquad \text{for } i, j \in \{1, 2\} : i \neq j, \tag{4.9}$$

where $T_{p_i}^{o}$ is the reflected position of the i^{th} robot in the world frame and $T_{r_j}^{o}$ is the odometry data from the other robot j. Other points different from the center could be used instead, such as the pivot points. However, a constant offset transformation would need to be taken into consideration in the previous equation. We can finally compare this reflected position to the actual position of each robot in order to determine the reflection synchronization error e_{p_i}

$$e_{p_i} = T_{p_i}^{o} * T_{r_i}^{o\,-1} \tag{4.10}$$

This error is then transformed to obtain $\dot{\boldsymbol{r}}_{ch}^{s}$, which is the desired linear and angular velocity of the chest. Since we are only interested in the horizontal velocity and angular velocity around the vertical axis for the chest frame, $\dot{\boldsymbol{r}}_{ch}^{s} \in \mathbb{R}^3$.

4.5.4 Visual Feedback

To further improve the synchronization, a second visual technique based on the optical flow observed from the robots cameras is added. We rely on the Lucas–Kanade method [Bouguet, 2000] to determine a sparse representation of the apparent motion between consecutive images of a video feed. However, in our case, since the robots need to be synchronized

and should always remain close from one another, we used the optical flow between a reference image and the video feed received from each robot, for the robots to maintain a fixed relative position with respect to each other.

The following assumption simplifies the image processing problem : instead of using a full image of the robot with the object and the (eventually cluttered) background, an ad-hoc mask is constructed to extract features in the reference image only on some relevant parts of the content, i.e., the robot. Note that the reference features are extracted only once, either offline or online at the initialization of the program, using the "Good Features To Track" [Shi et Tomasi, 1994] (GFTT) method to select the keypoints to track along the video frames. Another advantage is that the motion evaluated through the optical flow is absolute, i.e., it represents directly the error of synchronization instead of accumulating image-to-image motions (and drift) while keeping track of the errors. The mask and some extracted features are shown in Fig. 4.8.

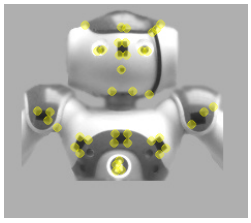


Figure 4.8 Mask used for visual feedback with features extracted by GFTT.

As one of the assumption of optical flow methods is that the motion is small between the two images, the search for corresponding points is limited to a neighborhood of the last detection, which serves as the initial estimate. Since the search is focused over a small window, it is more robust to blur and reduces the number of bad matches in other regions of the robot or on the background, compared to more traditional feature matching methods. This leads to less outliers, which in turn improves the speed and performance of most algorithms and geometrical verification (Least Median of Squares (LMEDS) [Rousseeuw, 1984], homography matrix computation and tests on reprojection errors). For points losses under occlusion, disturbance or rapid movements, we reset the estimate of these keypoints to the original mask features.

We transform the feature points in the reference image into approximate 3D coordinates in the camera frame at the reference position by :

$$\mathbf{P}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = d * K^{-1} * \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = d \begin{bmatrix} \frac{u_i - u_0}{f_u} \\ \frac{v_i - v_0}{f_v} \end{bmatrix} \tag{4.11}$$

where d is the (known) distance from the camera to the plane where the reference points are supposed to lie, and $K = \begin{pmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{pmatrix}$ is the camera intrinsic parameters matrix.

When tracking the points, we evaluate at each frame the homography H between the fixed 3D reference points $\mathbf{P}_i$ defined above and the tracked ones in frame t, $\mathbf{p}_i$, within a RANSAC scheme. Then we use the decomposition :

$$H \propto K \begin{bmatrix} R_1 & R_2 & t \end{bmatrix} \tag{4.12}$$

where R_1, R_2 are the first two columns of the 3D rotation matrix, and t the translation, to determine the camera localization [Zhang, 1998]. From the orthonormality of these vectors, we get a first estimate as :

$$\begin{bmatrix} R'_1 & R'_2 & t' \end{bmatrix} = \frac{K^{-1}}{\frac{1}{2}(\|R_1\| + \|R_2\|)} H \tag{4.13}$$

The third column R'_3 is found by taking the cross product between R'_1 and R'_2 to obtain the full 3D rotation matrix :

$$R' = \begin{bmatrix} R'_1 & R'_2 & R'_1 \times R'_2 \end{bmatrix} \tag{4.14}$$

Finally, to get the closest rotation matrix to R', as suggested in [Zhang, 1998], we perform a SVD decomposition on $R' = USV^T$ and set S to identity, to get :

$$R_{SVD} = UV^T \tag{4.15}$$

Beforehand, the singular values of S are checked to verify they are close to one with a 10% tolerance. When they differ too much from one, it means that the rotation found is probably wrong and must be discarded. Using U and V, the homogeneous matrix expressing the robot position is then :

$$T_c^r = \begin{bmatrix} UV^T & t' \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{4.16}$$

This gives us the relative positioning error between both robots. With the visual position computed, since the error found is absolute, it is transformed to obtain $\dot{\boldsymbol{r}}_{ch}^v$, which is the desired linear and angular velocity of the robot's chest. For the same reasons mentioned in Section 4.5.3, $\dot{\boldsymbol{r}}_{ch}^v \in \mathbb{R}^3$.

4.6 Control

Once a collision-free trajectory is found by the ARA* algorithm (Section 4.3), a set of footprints are defined along the trajectory [Rioux et Suleiman, 2015]. The second step is to define a Zero Moment Point (ZMP) trajectory. A trajectory of the CoM of the robot is then obtained using the preview control algorithm proposed in [Kajita *et al.*, 2003]. This algorithm, widely used in humanoid robotics, is simple to implement, yet efficient and yields a smooth CoM trajectory by minimizing the CoM jerk trajectory. The feet trajectories are obtained by spline interpolation between the footprints and the hands trajectories, and orientations are defined to minimize the walking swing effect and, in our case, to follow the object orientation. The output of the locomotion algorithm are $\dot{\boldsymbol{r}}_c$, $\dot{\boldsymbol{r}}_{lf}$, $\dot{\boldsymbol{r}}_{rf}$, which are the computed linear and angular velocity of the CoM, left foot, right foot, respectively.

4.6.1 Hierarchy of Tasks

To obtain the joint trajectories for each humanoid robot, a whole-body control scheme with prioritized tasks is formulated as follows :

$$
\begin{aligned}
&\min_{\dot{\boldsymbol{q}}} \dot{\boldsymbol{q}}^T \boldsymbol{Q} \dot{\boldsymbol{q}} \\
&\text{subject to} \\
&\text{First priority} \quad \begin{cases} \boldsymbol{J}_c\, \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_c \\ \boldsymbol{J}_{lf}\, \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_{lf} \\ \boldsymbol{J}_{rf}\, \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_{rf} \end{cases} \\
&\text{Second priority} \begin{cases} \boldsymbol{J}_{lh}\, \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_{lh} \\ \boldsymbol{J}_{rh}\, \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_{rh} \\ \boldsymbol{J}_p\, \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_p \\ \boldsymbol{J}_{ch}\, \dot{\boldsymbol{q}} = \alpha_1\, \dot{\boldsymbol{r}}^s_{ch} + \alpha_2\, \dot{\boldsymbol{r}}^v_{ch} \end{cases} \\
&\text{Joint velocity limits} \quad \hat{\dot{\boldsymbol{q}}}^- \leq \dot{\boldsymbol{q}} \leq \hat{\dot{\boldsymbol{q}}}^+
\end{aligned}
\tag{4.17}
$$

where $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ is the joint velocity vector to determine, n is the number of degrees of freedom, $\boldsymbol{Q}$ is a positive semi-definite weights matrix, $\boldsymbol{J}_c \in \mathbb{R}^{3\times n}$, $\boldsymbol{J}_{lf} \in \mathbb{R}^{6\times n}$, $\boldsymbol{J}_{rf} \in \mathbb{R}^{6\times n}$, $\boldsymbol{J}_{lh} \in \mathbb{R}^{6\times n}$, $\boldsymbol{J}_{rh} \in \mathbb{R}^{6\times n}$, $\boldsymbol{J}_p \in \mathbb{R}^{6\times n}$, $\boldsymbol{J}_{ch} \in \mathbb{R}^{3\times n}$ are the Jacobian matrices of the CoM, left foot, right foot, left hand, right hand, pelvis joint and chest, respectively. α_1 and α_2 are user-defined positive constants, such as $\alpha_1 + \alpha_2 = 1$. $\dot{\boldsymbol{q}}^-$ and $\dot{\boldsymbol{q}}^+$ are joint velocity limits.

The optimization problem given in Equation 4.17 can be transformed into a standard Quadratic Programming (QP) problem [Escande *et al.*, 2014; Rioux et Suleiman, 2015], which can be solved in real-time with an appropriate QP solver.

4.6.2 Dynamic Collision Avoidance and Replanning

It is important to check frequently for collision in case an obstacle has moved from its original position, the robots drift away from their planned trajectory, or the synchronization process of the robots requires a replanning of the whole trajectory. Hence, the future trajectory is always monitored for potential collisions with obstacles in the 2D occupancy grid. When a collision is foreseen, a replanning step is necessary and the trajectory is deformed as shown for a single robot, for the purpose of clarity, in Fig. 4.9. Even though, at first glance, the support polygon seems increased by including the robot-object-robot closed loop, the robots' support polygons are still defined by the contact between the feet and the ground. This is because the robots' arms are not fully bended, therefore the robots could fall forward or backward.

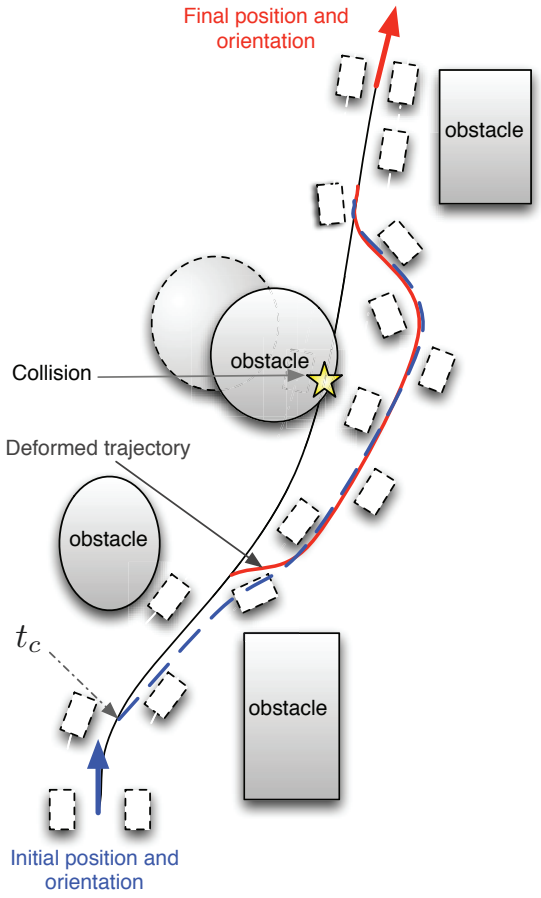


Figure 4.9 Replanning in case of collision detection : t_c is the instant at which a collision is foreseen, the new collision-free trajectory is in dashed-blue line, the deformed trajectory is in red line.

The new collision-free trajectory is found by the ARA* algorithm from the goal to the point at which the collision has been predicted. If the potential collision is due to drift and the environment has not changed, the Dijkstra grid does not need to be recalculated, which therefore greatly accelerating the replanning. As the humanoid robot walking pattern cannot be changed instantly, a time interval t_c is required to change the planned footprints. In the implementation of the ZMP preview control, a finite time horizon of two steps is used to compute the CoM trajectory. Therefore, if a collision is foreseen at instant t_c, the new collision-free trajectory provided by the ARA* algorithm is deformed to keep the next two footprints unchanged as shown in Fig. 4.9. The robot however stops if the deformed trajectory is in collision. Contrary to what Fig. 4.9 might suggest, two steps do not represent a significant distance. Since the robots are very small and the feet tend to slide, two steps are only a few centimetres or less.

4.7 Results

Experiments were conducted with Nao humanoid robots, manufactured by Aldebaran Robotics [Gouaillier *et al.*, 2009]. On top of the robots heads, we added an Asus Xtion Pro Live consumer-level depth camera (see Fig. 4.1). Six strips of black tape were added to the robots to increase contrast in low texture areas and make their detection for the visual synchronization (see Section 4.5.4) easier.

The object being transported, shown in Fig. 4.1, is a miniature table 600 mm long by 300 mm wide. The legs are cylinders 200 mm long and 40 mm in diameter. The legs are too short to touch the ground, so the table is fully supported by the robots.

4.7.1 Articulating the Arms

Without any hand position correction, the lateral swing causes large oscillations that are transmitted to the table and the load. To prevent this problem, the hands are controlled to follow stable trajectories using the whole-body control scheme, as explained in Section 4.6.

Without any correction, the average oscillation peak-to-peak movement of the hands is 47.6 mm. However, with the whole body control, the average hand distance from desired position has been reduced to 16.4 mm, reducing the hand error by 65.5%. As a result, significantly less oscillations are transmitted to the table, leading to a safer and enhanced carrying ability and load stability. Moreover, to minimize the impact of the backlash and the motors time response of the Nao robot, a PID controller on the robot joint positions has been implemented. By adding the PID controller with coefficient $K_p = 2, K_i = 0.01$ and $K_d = -0.015$, the error has been reduced further to 76.1%, for an average peak-to-peak movement of 11.4 mm.

The same tests were then conducted on the real robots and the real experiments results are consistent with simulation results. This still increases significantly the load stability. Also, note that the error cannot be completely cancelled for two main reasons. Firstly, the hand trajectories are second priority task only, which means that the robot respects those trajectories as far as the trajectories of first priority are fully followed. Secondly, the Nao robot has only 4 DOFs in each arm.

4.7.2 Visual Feedback

The mask used for the optical flow was created by taking a picture of one Nao from the other while they are standing in position with the table in hand, then by masking the background with a uniform gray color and finally blurring some edges to reduce false corner detection, especially at the bottom of the cropped robot image. Therefore, the mask is the same size as the original image and when compared to it and the optical flow represents an absolute error value since the robot is situated at the synchronized position in the mask. By using GFTT to find interest points, with a quality level of 0.07 and a minimum distance between features of 5 pixels, we have 41 features to track, as shown in Fig. 4.8.

To verify the precision and robustness of the optical flow for specific movements and static configurations, the following experiments were executed by moving the observed robot (the one being looked at). Meanwhile, the other (the observer, carrying the camera), was standing still. The motions made by the observed robot were linear motions in $\pm$X and $\pm$Y and rotations around the Z axis (Yaw).

The results in Table 4.2 show that the Y axis error is very low and consistent, while the X axis and rotation around Z axis (Yaw) errors are not that precise. On a 2D image, in our case YZ-plane, the displacement of a pixel along an image axis is much easier to detect. It means that the Y, Z and Roll motions of the robots are determined with more precision and consistency than the X, Pitch and Yaw motions. In the X axis, motions seem to be detected further away than they really are. The Yaw estimation is very inaccurate, its use for the moment is then limited as an indicator for error to correct rather than an absolute error value to use as feedback.

However, these static tests do not contain any motion blur, which is usually an important problem with moving and oscillating humanoid robots. An example of this phenomenon is illustrated in Fig. 4.10. This image is taken from the experiments performed in Section 4.7.3 : the black lines are the features found and matching the computed homography, while the white lines are found matches that do not fit the homography model. To determine which points are valid, a Least Median of Squares (LMEDS) on the re-projection error is used. The figure illustrates the robustness of our technique to heavy blur.

4.7.3 Navigating in a Cluttered Environment

To test the system as a whole and to validate the proposed algorithms, we conducted a series of five experiments. In each experiment, the robots starting and goal positions were

Tableau 4.2 Static visual feedback

	Motion error	Observed error	Unmatched features (%)	Rejected by homography (%)	Reprojection error (x,y) (m)
Moving robot perception					
Move +X	0.05 m	$-0.044 \pm 14.91\,10^{-3}$	1.2	26.7	$(-0.28\,10^{-3}, 0.99\,10^{-3})$
Move -X	0.05 m	$0.051 \pm 19.83\,10^{-3}$	7.9	24.0	$(0.79\,10^{-3}, 1.67\,10^{-3})$
Move +Y	0.1 m	$-0.098 \pm 7.38\,10^{-3}$	2.4	18.8	$(-2.77\,10^{-3}, 7.31\,10^{-3})$
Move -Y	0.1 m	$0.099 \pm 0.05\,10^{-3}$	4.9	28.2	$(2.62\,10^{-3}, 2.08\,10^{-3})$
Turn +Z rad	0.17 rad	$-0.04 \pm 1.02\,10^{-3}$	9.8	18.9	$(0.77\,10^{-3}, -1.83\,10^{-3})$
Turn -Z rad	0.17 rad	$0.26 \pm 2.01\,10^{-3}$	13.4	26.7	$(2.79\,10^{-3}, -4.00\,10^{-3})$
Idle robot perception					
Move +X m	0.05 m	$-0.041 \pm 6.46\,10^{-3}$	3.7	20.2	$(-0.01\,10^{-3}, 0.07\,10^{-3})$
Move -X m	0.05 m	$0.058 \pm 2.29\,10^{-3}$	4.9	15.8	$(-0.61\,10^{-3}, -0.62\,10^{-3})$
Move +Y m	0.1 m	$-0.103 \pm 1.42\,10^{-3}$	12.2	14.2	$(-4.38\,10^{-3}, -0.63\,10^{-3})$
Move -Y m	0.1 m	$0.100 \pm 0.17\,10^{-3}$	3.6	26.6	$(-0.52\,10^{-3}, 3.68\,10^{-3})$
Turn +Z rad	0.17 rad	$0.372 \pm 16.72\,10^{-3}$	18.6	19.3	$(-1.83\,10^{-3}, 2.79\,10^{-3})$
Turn -Z rad	0.17 rad	$-0.325 \pm 34.57\,10^{-3}$	6.0	22.0	$(-7.29\,10^{-3}, 2.70\,10^{-3})$

Figure 4.10 Demonstration of the robustness for highly blurry images

chosen in such a way that the robots had to navigate among objects on the ground. The objects served as obstacles to be avoided by the robots and the transported object. They were placed to form various feasible paths and force tight turns to take advantage of the additional degrees of freedom (the rotations of the object θ_{ob} and θ_{r_2}).

Fig. 4.11 shows the waypoints for each type of experiment. The start and end of each experiment are all pairs of consecutive waypoints. In Fig. 4.12, the obstacles are in yellow, while the red areas around them are inflation zones where the cost is higher than in free space, to prevent the robots from passing too close to obstacles. These zones are used as a security buffer. The centers of the robots and the object should avoid, if possible, to generate plans passing inside this zone. The cyan zone is a forbidden zone, because if the center of the object or the robots enters it, it means that an edge is in collision with an obstacle.

The ARA* planner parameter initial value $\epsilon = 3$ means that the suboptimal solution cannot be worse than three times the optimal solution cost. A time limit of 5 seconds was chosen and within that time, ϵ was successfully decreased to 1 on every run, which corresponds to the optimal solution. For each generated path, we measured various data about the planner and the trajectories. These results are summarized in Table 4.3.

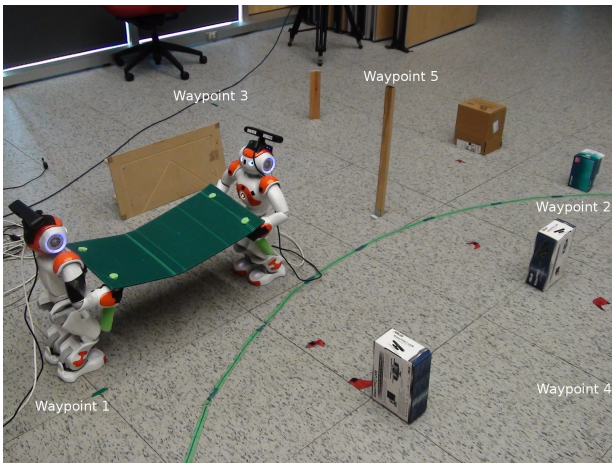


Figure 4.11 Picture of the obstacles, robots and waypoints.

Tableau 4.3 Statistics about the experiments

	Means	Std. dev.
Total time (s)	94.28	18.14
Trajectory length (m)	2.27	0.19
Average velocity (m/s)	0.025	0.003
Initial solution time ($\epsilon = 3$) (ms)	12	13
Optimal solution time ($\epsilon = 1$) (ms)	170	68
Initial Nodes Expansions	195.6	161.3
Total Number Of Expansions	3346.0	1674.3
Total Actions in Path	158.4	10.3

Even though in our case the trajectories are quite small, the use of sub-optimal solutions has proven to be significantly faster. Indeed, for our experimental settings, it is about 14.17 times faster and takes 17.1 times less states expansions to compute the solution for $\epsilon = 3$ as opposed to the optimal solution. This could be especially useful for real life large scale distances and experiments where quick reactions and planning are necessary.

In Table 4.4, the errors in X, Y and Yaw for both robots are shown. During each experiment, the reflection error is measured at 10 Hz while the visual information is registered as fast as possible (about 14.5 Hz) when the computed homography is valid.

Tableau 4.4 Synchronization errors

	X (m)	Y (m)	Yaw (rad)
Reflection Error	$3.44\,10^{-4}$	$-6.70\,10^{-3}$	$3.00\,10^{-6}$
Reflection Error Std	$1.54\,10^{-2}$	$3.29\,10^{-2}$	$3.50\,10^{-2}$
Visual Feedback Error	$6.29\,10^{-3}$	$-2.19\,10^{-2}$	$4.78\,10^{-2}$
Visual Feedback Error Std	$4.19\,10^{-2}$	$3.13\,10^{-2}$	$1.40\,10^{-1}$

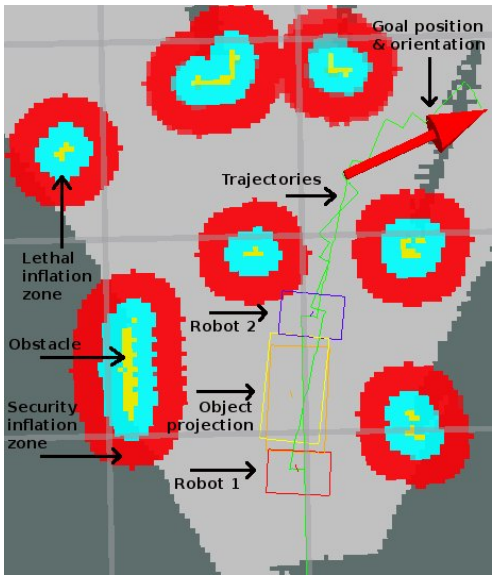


Figure 4.12 Map of the starting and ending positions with obstacles, lethal and security inflation zones around them, the Nao and object footprints and goal position/orientation.

For all values with the exception of the visual axis Y, all the error coefficients converged to zero. This exception can be explained by the ambiguity that takes place when distinguishing between the observed robot translating in place and the observer robot rotating in place. This distinction is taken into account by using the odometry. However, if the error is due to drift, thus not correctly reported by the odometry, an error can occur. Here, with an error of -0.0152m in Y, at a distance between the cameras of the robots of 0.67m, a drift of only 1.28 degrees can explain the difference.

The high standard deviations are caused mainly by the natural oscillation of the robots while walking. Particularly in the Y axis, both techniques detected the same lateral motion of the robot. In the X axis, it is also because it is the main direction of movement and the system wants to follow the planned trajectories as a main priority. The focus here is to reduce the errors, but not necessarily to cancel them completely if it means that it will prevent normal motion or highly impair it. The visual feedback error is even higher in X because the torso and head of the robots oscillate in this axis while walking. This oscillation is not detected by the reflection, as it uses the ZMP of the robot. The Yaw from the reflection is very small, principally because they are physically constrained by the object. However, Yaw error detected visually is high in comparison. As previously discussed, this value is more an indication of the direction and magnitude of the movement because it is hard to compute precisely using only 2D features, thus making it not very accurate.

In addition to the visual errors measured from the Naos' camera, the quality and robustness indicators previously discussed were computed and monitored. They are summarized in Table 4.5.

Tableau 4.5 Visual feedback errors

	Unmatched features (%)	Rejected points (%)	Reprojection errors (x,y) (m)	Processing time (ms)
Error	5.5	25.2	$(5.49\,10^{-4}, 6.02\,10^{-4})$	68.9
Error Std	4.3	6.4	$(3.14\,10^{-3}, 3.12\,10^{-3})$	10.5

It can be observed that the monitored parameters are in line with what was measured statically, even though heavy motion blur was added in the experiments. On one hand, the number of unmatched features is in the lower range of the static data. It is caused by the automatic resetting of invalid features that can more easily reattach themselves to a valid point when moving throughout the image than when staying at an invalid position. On the other hand, the rejected points are in the upper range of the static equivalent. The cause is motion blur, as well as more varying background, that can both cause wrong matches.

Robustness and effectiveness of the method while in motion is demonstrated as it successfully matches and fits the majority of points even during large movements, with limited framerate and camera resolution.

The output from the SLAM library, RTAB-Map, are shown in Fig. 4.13. The localization information is displayed as a continuous colored line and the mapping information is represented by the rest of the incrementally built point cloud. This map could be used for any other experiment taking place at the same location, for quick localization in a known environment. It is important to note that the ground in the testing room had sufficient texture and patterns to produce reliable data for the SLAM library. When tested on a plain ground, RTAB-Map could not, however, extract enough features for visual localization using only the obstacles.

With the table linking the two robots together, significant drift caused by the visual odometry imperfection can make the Nao drift when navigating as shown in Fig. 4.14. This drift is mainly due to the cameras not being fixed rigidly enough on the Nao heads and slowly changing position over time. A special rigid head mount should greatly reduce this drift.

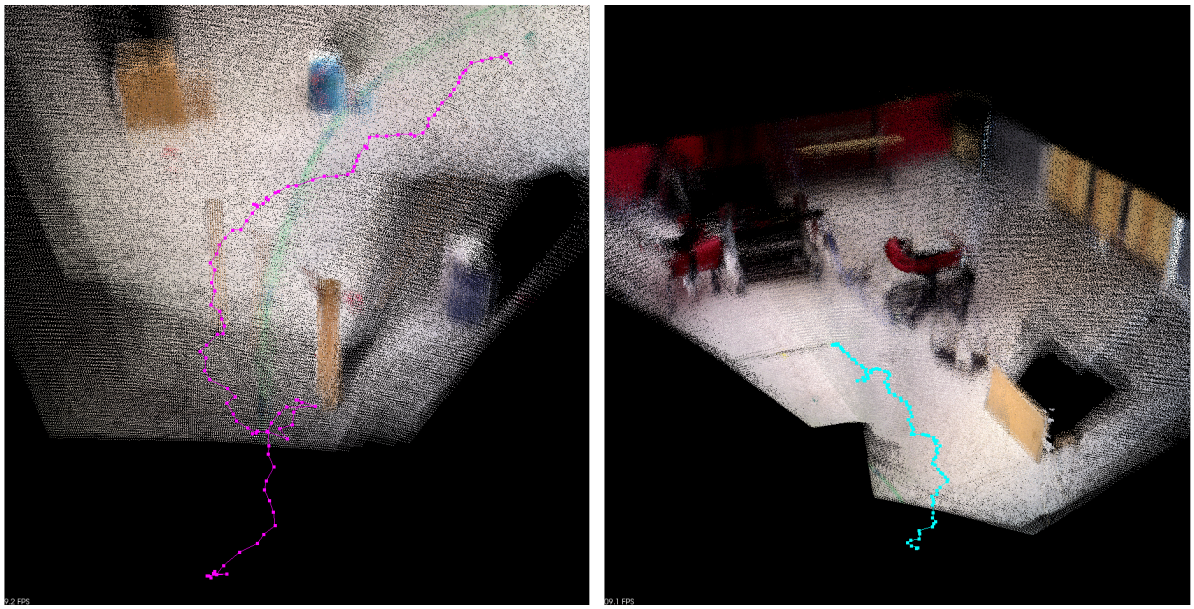


Figure 4.13 Left : localization (purple line) and mapping (point cloud) close up for the backing robot. Right : Localization (cyan line) and full mapping (point cloud) for the forward robot.

4.8 Conclusion and Future Work

In this paper, we have presented a system capable of carrying a long table with two humanoid robots while navigating in a cluttered environment. We also gave practical insights into the implementation of the approach on a real humanoid robot. Our method uses a lattice-based planner designed for two robots transporting and articulating an object. This includes a reduction of the state dimensionality, a choice of adequate simple and complex primitives and a cost function computed while considering these restrictions.

When moving throughout the environment, a depth camera and a SLAM library map the obstacles in real-time and provide a visual odometry. This information is then fused with each robot odometry to provide a consistent, continuous and reliable odometry data. While mapping the environment, dynamic collisions are foreseen and trajectories modified while in motion. A finite horizon of two steps is however necessary before the robot can change its current trajectory. It would be possible to integrate the SLAM, the 3D occupancy grid and the loop closure algorithms provided by RTAB-Map with the memory efficient OctoMap [Hornung *et al.*, 2013] for planning and obstacle detection. However, since the project is mainly in 2D for the moment, this integration was not performed.

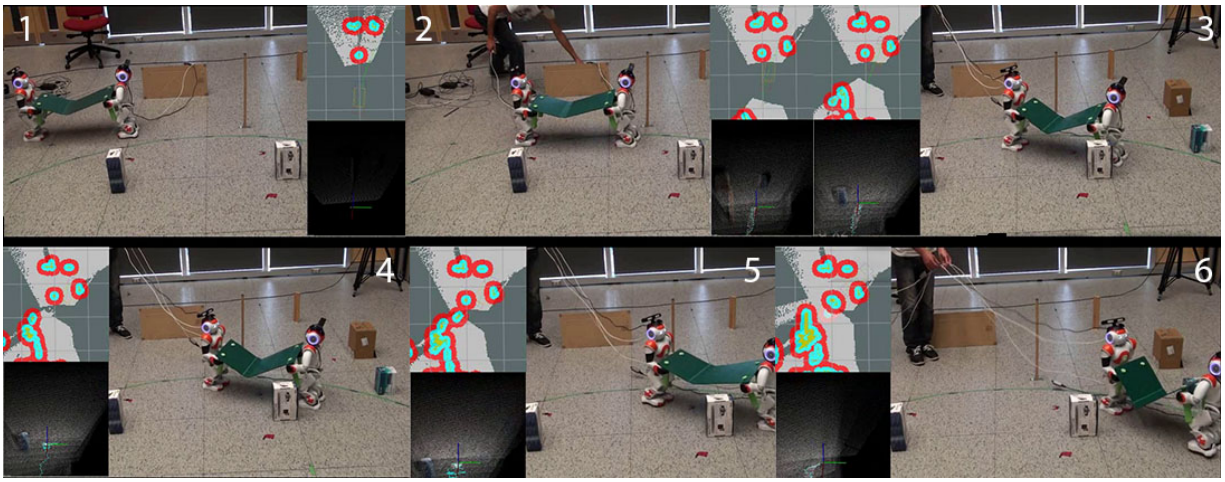


Figure 4.14 Snapshots of the Nao robots navigating with an object

Moreover, by controlling the hands adequately with a whole-body control scheme coupled with a PID, we could maneuver the object in tight turns, reduce significantly the lateral swing and avoid its propagation to the object and between robots.

In future works, to make the robots completely autonomous, it is necessary to implement the algorithms entirely on the Nao itself to be processed by its internal CPU. The biggest implementation challenge is the platform limited processing power in comparison with other high-end humanoid robots that have often multiple onboard computers. For example, when the visual feedback algorithms were implemented on the Nao robots themselves, the frame rate was reduced to 1 fps on average, which makes it too inaccurate to use. To further improve the synchronization speed and make the system behavior more human-like, the arms should be used to absorb part of the error instead of only controlling the object for special primitives and swing reduction. In addition, the visual feedback we use for synchronization could be used to determine the drift occurring between the robots. Since they are able to see each other's real position, the visual odometry drift could be corrected to improve the long term navigation. Also, a slippage avoidance and turning strategies such as the ones proposed in [Li *et al.*, 2015] and [Yeon et Park, 2014] would be investigated, along with a motion planning which considers a 3D model of the environment [Harada *et al.*, 2010].

CHAPITRE 5

Conclusion

Dans ce mémoire, nous avons proposé un système capable de transporter un grand objet à l'aide de deux robots humanoïdes. Le système composé des deux robots et de la table forme une chaine fermée articulée aux points milieux situés entre les mains de chaque robot. Ces articulations ont pour but de réduire la grosseur de l'empreinte du système et de le rendre plus agile et flexible lorsque celui-ci se déplace dans un environnement encombré. Pour ce faire, un planificateur de haut niveau de type ARA* qui utilise des primitives de mouvements permet un contrôle précis de la chaine fermée.

Afin de se déplacer dans un environnement encombré sans entrer en collision avec des obstacles, des caméras RGB-D ont été ajoutées sur la tête des robots afin de générer une carte détaillée de l'environnement en temps réel. Pour se faire, la librairie de SLAM RTAB-Map [Labbe et Michaud, 2014b] a été utilisée. Celle-ci permet également de fournir une odométrie visuelle relativement robuste. Puis, pour pouvoir davantage améliorer cette information, elle est fusionnée avec l'odométrie interne des robots pour obtenir des données continues et fiables.

Puisque le système en boucle fermée doit rester stable et en équilibre tout au long de la navigation, plusieurs composants de synchronisation ont été mis en place afin de réduire le plus possible les forces affectant chaque robot. Tout d'abord, les mains sont contrôlés par un système de contrôle du corps complet jumelé avec un PID afin de diminuer l'impact du balancement latéral causé par la marche. Ensuite, les robots suivent leurs trajectoires segmentées respectives, tout en se mettant d'accord sur la progression de ces sections. Puis, le contrôleur de mouvement central est modifié pour prendre en considération l'erreur entre la position actuelle de chaque robot et leur position désirée par rapport à leur projection dans la chaine robot-object-robot. Finalement, un retour visuel de la position des robots face-à-face est utilisé pour trouver le déplacement relatif de ceux-ci et corriger les erreurs de positions et rotations détectées.

Les expériences réelles menées sur des robot humanoides Nao montrent bien la performance des algorithmes développés dans des situations réalistes. Avec seulement un seul robot et un chariot de fortune, nous avons montré que le poids maximal pouvant être transporté par le Nao est au moins 11 fois plus grand que celui par le robot seul. Nous avons aussi démontré

que deux robots humanoides identiques peuvent se déplacer de manière synchronisée afin de transporter un object de grande taille, qu'un seul robot ne pourrait transporter seul.

Les contributions apportées par ce projet de maîtrise sont les suivantes :

- Un système de contrôle de corps complet qui permet à un robot humanoïde d'utiliser ses mains et ses bras pour contrôler les mouvements d'un système robot-objet-robot (e.g. virage serré) ;

- Une approche sans capteur pour sélectionner automatiquement le jeu approprié de primitives en fonction du poids de la charge ;

- Une technique de filtrage efficace pour ignorer le chariot et le poids du champ de vue du robot tout en améliorant les performances générales des algorithmes de SLAM ;

- Un algorithme de planification de mouvement avec un chariot et avec deux robots humanoïdes pour trouver une trajectoire sans collision en utilisant une carte construite en temps réel ;

- Une odométrie continue et fiable formée de la fusion d'une odométrie visuelle et de l'odométrie interne du robot ;

- Un système de synchronisation de mouvement utilisant le décalage relatif des robots par rapport à la trajectoire planifiée, la projection des robots en passant par le positionnement des mains et l'erreur en position et rotation déterminées visuellement par les robots face-à-face.

Pour des travaux futurs, afin de rendre ce projet complètement autonome, il sera nécessaire de l'implémenter entièrement sur le Nao lui-même pour être calculé par le CPU embarqué. Le plus gros défi à surmonter pour réaliser ceci est la faible capacité de calcul des processeurs des robots Nao en comparaison aux autres robots totalement autonomes plus avancés qui ont généralement plusieurs ordinateurs embarqués. Les solutions possibles sont donc soit de changer les pièces physiques afin d'augmenter la puissance de calcul du robot utilisé, soit de modifier les algorithmes pour qu'ils soient plus efficaces et optimisés pour la plateforme actuelle. La partie du système la plus coûteuse en termes de calculs est sans aucun doute les processus traitant l'odométrie visuelle. En utilisant différents paramètres dans la librairie RTAB-Map afin de réduire la précision et la fréquence de la génération de carte et de la localisation, il serait possible d'améliorer les performances en termes de temps de calculs, mais avec également des impacts négatifs sur les résultats.

Puis, une stratégie de réduction de glissement similaire à celles proposées dans [Li *et al.*, 2015] et [Yeon et Park, 2014] pourrait être utilisée.

Finalement, le passage vers un environnement perçu en 3D rendrait le projet plus versatile [Harada *et al.*, 2010]. En effet, dans son état actuel, les obstacles étant projetés sur le sol pour créer une carte d'occupation 2D, un partie de l'information est perdue et l'approche empêche de se déplacer à certains endroits qui pourraient être autrement valides. Puisque RTAB-Map peut déjà être utilisé en 3D, l'intégration du très connu Octomap [Hornung *et al.*, 2013] permettrait de gérer rapidement de grandes cartes 3D. De cette façon, en plus d'être autonomes, les robots pourraient se déplacer dans tous types d'environnements sans problèmes.

ANNEXE A

Articles de conférence

2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Congress Center Hamburg
Sept 28 - Oct 2, 2015. Hamburg, Germany

Humanoid Navigation and Heavy Load Transportation in a Cluttered Environment

Antoine Rioux and Wael Suleiman

Abstract— **Although in recent years several studies aimed at the navigation of robots in cluttered environments, just a few have addressed the problem of robots navigating while moving a large or heavy object. This is especially useful when transporting loads with variable weights and shapes without having to change the robot hardware. On one hand, a major advantage of using a humanoid robot to move an object is that it has arms to firmly grasp it and control it. On the other hand, humanoid robots tend to have higher drift than their wheeled counterparts as well as having significant lateral swing while walking, which propagates to anything they carry. In this work, we present algorithms for a humanoid robot navigating in a cluttered environment while pushing a cart-like object. In addition, the algorithms make use of the hands and arms to articulate the cart when executing tight turns using whole body control scheme to reduce the lateral swing effect on the load and ensure a safe transport. Experiments conducted on a real Nao robot assessed the proposed approach and algorithms, they show that the payload of a humanoid robot can be significantly increased without changing the humanoid robot's hardware, and therefore enact the capacity of humanoid robots in real-life situations.**

I. Introduction

One of the advantages of having arms on a robot is that it can carry a load. This capacity can be useful for a wide range of actions, including transporting objects from one place to another. However, the maximum payload is generally pretty low and generates a lot of instability if it is held at the arm's length. While it is possible to increase the strength of the motors in the legs and arms, it is not the best solution since a motor's power is proportional to its size, weight and price. Instead of putting the entire load directly on the robot, a cart-like object can be used to help supporting the weight. The cart can then be pushed by the robot and moved around more easily without having to modify the robot's hardware to fit the load.

Many researches have been done on navigating robots in a cluttered environment, but adding a controllable object supporting a heavy load and navigating it safely has not been widely examined. Furthermore, using a humanoid robot, which has unstable balance and great motion swinging, increases the difficulty of the task in hand. The objective is to plan a stable and safe trajectory that uses a whole body control scheme.

To address the problem of navigating a cart-like object supporting a load with a humanoid robot in a cluttered environment, the following sub-problems should be solved:

Antoine Rioux and Wael Sulieman are with Electrical and Computer Engineering Department, the Faculty of Engineering, University of Sherbrooke, Sherbrooke, Canada {antoine.rioux, wael.suleiman}@usherbrooke.ca

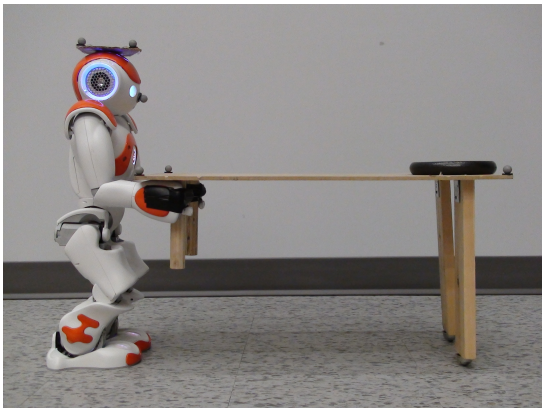


Fig. 1. The Nao robot holding the cart-like object

(I) planning, (II) sensing and (III) controlling the cart-like object. To plan a safe trajectory between obstacles, an anytime search-based planner exploits a given set of motion primitives that consider both the robot and the cart footprint. The second step is to find the humanoid robot's footprints, the humanoid's feet and hands trajectories are then computed in order to minimize the swing effect and follow the cart trajectory using a task priority whole body control scheme.

The main contribution of the paper is providing a framework for humanoid navigation in a cluttered environment while manipulating a cart-like objet that carries a heavy load.

This paper is organized as follows. Section 2 presents an overview of related works. Section 3 describes the robot-cart model and the implemented algorithms. In Section 4, simulation and real results are presented and analyzed.

II. Related work

Many studies have been done on robots moving objects to a specific goal. Most of them though are executed with multiple wheeled robots that position themselves around the object to push it in the desired direction. One of the first examples of this behavior for a movable target tried to reproduce how ants move a prey bigger than them by trial and error [1]. Another added the possibility of pulling the object by using a rudimentary one degree of freedom arm [2], while [3] have the additional benefit of being able to manipulate an object in 3D when necessary, instead of 2D. In these researches, the manipulator comes in contact with the object at only one point, which barely allows any control over the object while moving and manipulating it. Holonomic wheeled robots are indeed less complex to control than humanoids, they are mainly useful at sliding box-like object

978-1-4799-9993-4/15/$31.00 ©2015 IEEE

on the ground. Therefore, a robot with humanoid arms allow a better control of the structure of an object and is more suitable to control cart-like objects with loads.

Other works have explored the usage of humanoid robots that push objects while keeping a firm grip on the handles. In theory, two arms are enough to fully constrain and control all the degrees of freedom of a cart. It was demonstrated that Honda's ASIMO is capable of moving a large cart in rooms and hallways [4]. However, the cart is mostly controlled as a Dubins car instead of taking advantage of the full possibility of the humanoid robot's holonomic movement. Also, the arms serve only as a mean of attaching the cart to the robot and are not taken into consideration to control the cart further.

Another project used a biped robot to push a person on a wheelchair [5]. To keep balance while performing the desired task, this HRP-2 and the ASIMO both implement an approach with a Zero-Moment Point (ZMP) offset. Although the HRP-2 has a heavy load to move on the wheelchair, the possible movements are akin to those of the ASIMO robot as the cart is controlled like a Dubins car. Furthermore, none of these examples minimize the footprint of the robot-cart contraption by turning the object because they are considered as one static bloc. In a tight and cluttered environment, taking advantage of the arms to manipulate the cart-like object is beneficial and possibly necessary.

In [6], a planning method for humanoids to navigate among movable obstacles has been proposed. The main purpose of that method is to find a path from a starting to a goal points in a complex environment where the robot can easily move objects to create a clear path, if it exists one. Our objective is however different, as we are interested in not only navigating in a cluttered environment, but also transporting a heavy load that is significantly bigger than the humanoid payload.

In the work of [7], a PR2 robot possessing a wheeled holonomic base and two 7 DoF (Degree of Freedom) arms is used to push a small cart and control its orientation. However, despite having humanoid arms to orient the cart, since the PR2 has wheels instead of legs, no lateral swing is transmitted to the transported object causing oscillations and instability, which is a problem with humanoid robots. Furthermore, the cart is very small and light weight with respect to the PR2, in contrast to our proportionally big cart that is able to carry a load heavier than the robot itself.

III. ROBOT-CART MODEL AND ALGORITHMS

A. Path finding

To be able to navigate trough a cluttered environment, a path provided by a motion planning algorithm is essential. Among the different possibilities, we chose a lattice-based graph planning with an ARA* search [8]. This choice is mainly motivated by the use of motion primitives that assures feasible robot-cart configurations and transitions. The environment is modeled by a 2D grid costmap that discriminates obstacles from free space at a fixed threshold and allow obstacles inflations to increase the security margin.

Each node of the search graph needs a complete state representation of the robot-cart to properly operate. To achieve this, it is possible to model the state in $\mathbb{R}^2 \times \mathbb{S}^1 \times \mathbb{R}^2 \times \mathbb{S}^1$:

$$s = (x_r, y_r, \theta_r, x_{ca}, y_{ca}, \theta_{ca}) \tag{1}$$

Where x_r, y_r and θ_r are the positions and orientation of the robot, x_{ca}, y_{ca} and θ_{ca} are those of the cart. As the working space of our robot's arms is too small to fully take advantage of both rotation and translation, the problem can be simplified by setting a pivot point positioned in the middle of both hands to reduce the dimensionality of the search space, the chosen position of the pivot point also maximizes the rotation range within the robot's workspace (see **pivot point 1** in Fig. 5), resulting in a 4 dimensions state space $\mathbb{R}^2 \times \mathbb{S}^1 \times \mathbb{S}^1$:

$$s = (x_r, y_r, \theta_r, \theta_{ca}) \tag{2}$$

Even-though the above simplification removes the ability of the cart to translate on the plane, the robot retains enough manipulability to minimize the cart footprint on tight turns.

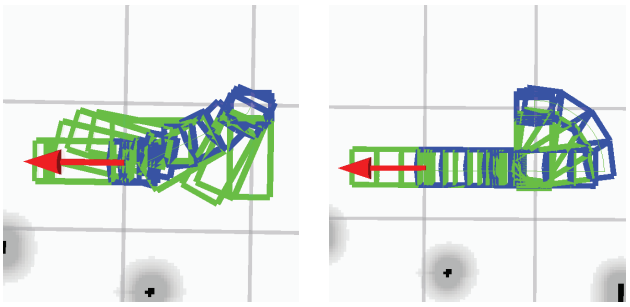
In a lattice-based graph planner, the transition between the nodes is a discrete action chosen within a fixed-set of possible actions called motion primitives. An important feature of the lattice representation is that each of those connections is a feasible path, in contrast to other forms of graph search, including "4-connected" or "8-connected" grid. This makes it really suitable for highly constrained systems, such as a robot moving a cart.

Because the cart can carry different loads, multiple sets of primitives are needed depending on the load's weight. Without load, the robot is holonomic and can move in any direction. A subset of movements composed of forward, backward, diagonal, rotate in place and turn while moving forward is used to reduce planning time while focusing on forward movements. With a heavy load though, moving sideways and rotating in place becomes really difficult because of the increased friction. For this reason, when the weight becomes too important, rotation occurs around a pivot point situated between the two table's legs touching the ground (see **pivot point 2** in Fig. 5). Thus, changing the feasible primitives is necessary for the lattice representation to remain coherent. An example of right turn for the omnidirectional and the heavy load sets of primitives are presented in Fig. 2.

The cost function of a transition from state s to s' is based on the time to execute that transition and is computed as follows:

$$Cost = \begin{cases} \frac{\sqrt{(\Delta x_r)^2 + (\Delta y_r)^2}}{\dot{\boldsymbol{r}}^+} \times DF & \text{if } \Delta x_r \neq 0 \text{ or } \Delta y_r \neq 0 \\ \frac{\Delta \theta_r}{\dot{\theta}^+} \times DF & \text{otherwise} \end{cases} \tag{3}$$

where Δx_r, Δy_r and $\Delta \theta_r$ are the differences between the x_r, y_r and θ_r, which are the coordinates of the robot's pelvis joint, between states s and s', DF is a difficulty factor associated with each primitives, $\dot{\boldsymbol{r}}^+$ is the maximal robot linear velocity and $\dot{\theta}^+$ is the maximal angular velocity for turning in place. The Euclidean distance between both states is computed and then divided by the maximum velocity



(a) omnidirectional primitives set (b) heavy load primitives set

Fig. 2. A right turn executed by the robot (blue) pushing the cart (green) to the goal (red arrow) with both sets of primitives.

of the robot in the direction of the movement to give the approximate time to execute the primitive.

For both instances, the difficulty-factor associated with each primitive is then multiplied by the time cost. This DF is used to prioritize or penalize certain motions or directions, which result in a smoother and a more natural looking trajectory. For example, turning in place then moving forward takes a longer time than moving in diagonal. However, on a long distance, the former reduces the trajectory footprint and is therefore more natural looking while reducing the chances of drifts caused by the table movements. For those reasons, moving sideway has a higher DF than turning and moving forward.

A* is one of the most popular search method at finding a solution path using a cost function. In addition to the cost function, a heuristic bias the search towards the most promising states. In our case, our heuristic is a 2D grid containing all the Dijkstra distance costs from the start to the goal states. Even though A* is optimal when it finds a solution, that solution may not always exists or cannot be found within a certain time limit. The Anytime Repairing A* (ARA*) planner focuses on delivering a suboptimal solution as fast as possible, this solution is then optimized iteratively to obtain the optimal solution.

B. Humanoid footprints and whole body control scheme

Once a collision-free trajectory is found by the ARA* algorithm, a set of footprints are defined along the trajectory as it is shown in Fig. 3. Even-though, at first glance, the support polygon appears to be increased by adding the cart, the robot's support polygon is always defined by the contact between the feet and the ground. This is because the robot's arms are not fully bended, therefore the robot could fall forward or backward .

The second step is to define a Zero Moment Point (ZMP) trajectory. A trajectory of the Center of Mass (CoM) of the robot is then obtained using the preview control algorithm proposed in [9]. This algorithm has been widely used by researchers in humanoid robotics, it is simple to implement, yet efficient and yields a smooth CoM trajectory by minimizing the CoM jerk trajectory. The feet trajectories are obtained by spline interpolation between the footprints and the hands trajectories and orientations are defined in order to minimize the walking swing effect as well as follow the cart orientation.

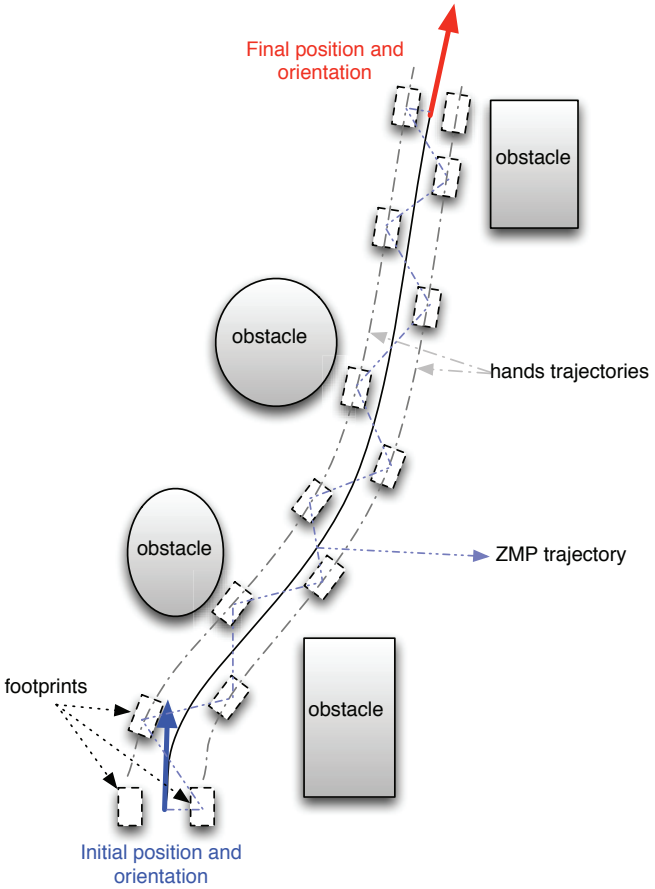


Fig. 3. Overview of the motion planning procedure

To obtain the humanoid robot's joint trajectories, a whole body control scheme with prioritized tasks is formulated as follows:

$$
\begin{aligned}
&\min_{\dot{\boldsymbol{q}}} \dot{\boldsymbol{q}}^T \boldsymbol{Q} \dot{\boldsymbol{q}} \\
&\text{subject to} \\
&\text{First priority} \quad \begin{cases} \boldsymbol{J}_c\, \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_c \\ \boldsymbol{J}_{lf}\, \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_{lf} \\ \boldsymbol{J}_{rf}\, \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_{rf} \end{cases} \\
&\text{Second priority} \quad \begin{cases} \boldsymbol{J}_{lh}\, \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_{lh} \\ \boldsymbol{J}_{rh}\, \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_{rh} \end{cases} \\
&\text{Joint velocity limits} \quad \tilde{\dot{\boldsymbol{q}}}^- \leq \dot{\boldsymbol{q}} \leq \tilde{\dot{\boldsymbol{q}}}^+
\end{aligned}
\tag{4}
$$

where $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ is the joint velocity vector, $\boldsymbol{Q}$ is a positive semi-definite matrix, $\boldsymbol{J}_c \in \mathbb{R}^{3\times n}$, $\boldsymbol{J}_{lf} \in \mathbb{R}^{6\times n}$, $\boldsymbol{J}_{rf} \in \mathbb{R}^{6\times n}$, $\boldsymbol{J}_{lh} \in \mathbb{R}^{6\times n}$, $\boldsymbol{J}_{rh} \in \mathbb{R}^{6\times n}$ are the jacobian matrices of CoM, left foot, right foot, left hand and right hand respectively. $\dot{\boldsymbol{r}}_c$, $\dot{\boldsymbol{r}}_{lf}$, $\dot{\boldsymbol{r}}_{rf}$, $\dot{\boldsymbol{r}}_{lh}$, $\dot{\boldsymbol{r}}_{rh}$ are the linear and angular velocity of CoM, left foot, right foot, left hand and right hand respectively.

$\tilde{\dot{\boldsymbol{q}}}^-$ and $\tilde{\dot{\boldsymbol{q}}}^+$ are generalized joint velocity limits defined as follows:

$$\begin{aligned}\tilde{\dot{\boldsymbol{q}}}_j^+ &= \begin{cases}\zeta\frac{(q_j^+ - q_j)-q_s}{q_i-q_s} & \text{if } q_j^+ - q_j \leq q_i \\ \dot{q}_j^+ & \text{otherwise}\end{cases} \\ \tilde{\dot{\boldsymbol{q}}}_j^- &= \begin{cases}-\zeta\frac{(q_j - q_j^-)-q_s}{q_i-q_s} & \text{if } q_j - q_j^- \leq q_i \\ \dot{q}_j^- & \text{otherwise}\end{cases}\end{aligned} \tag{5}$$

where $\tilde{\dot{\boldsymbol{q}}}_j^*$ is the j element of the vector $\tilde{\dot{\boldsymbol{q}}}^*$, q_j is the value of joint j, q_j^+ and q_j^- are the upper and lower limits for the joint j, ζ, q_i and q_s are user-defined positive constants, q_i is usually called the interference distance. It can be easily proven that the equalities constraints in (5), not only yield a motion within the humanoid's velocity limits, but also the joints limits are respected as well with a safety margin equals to q_s:

$$q_j^- + q_s \leq q_j \leq q_j^+ - q_s$$

Eq. (5) provides a compact and efficient way for dealing with both of velocity and joint limits, it has been originally proposed in [10].

The optimization problem (4) can be efficiently approximated by the following standard Quadratic Programming (QP) problem:

$$\begin{aligned}&\min_{\boldsymbol{X}} \boldsymbol{X}^T \boldsymbol{H} \boldsymbol{X} \\ &\text{subject to} \\ &\boldsymbol{J}_1 \boldsymbol{X} = \dot{\boldsymbol{r}}_1 \\ &\boldsymbol{J}_2 \boldsymbol{X} = \dot{\boldsymbol{r}}_2 \\ &\boldsymbol{X}^- \leq \boldsymbol{X} \leq \boldsymbol{X}^+\end{aligned} \tag{6}$$

Where:

- $\boldsymbol{X} = \begin{bmatrix}\dot{\boldsymbol{q}} \\ \boldsymbol{\delta}_1 \\ \boldsymbol{\delta}_2\end{bmatrix}$, $\boldsymbol{\delta}_1 \in \mathbb{R}^{15}$ and $\boldsymbol{\delta}_2 \in \mathbb{R}^{12}$ are slack variables.
- $\boldsymbol{H} = \begin{bmatrix}\boldsymbol{Q} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{Q}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{Q}_2\end{bmatrix}$, $\boldsymbol{Q}_1 \in \mathbb{R}^{15\times 15}$ and $\boldsymbol{Q}_2 \in \mathbb{R}^{12\times 12}$ are user-defined positive definite matrices. In order to respect the task priority, the following condition should be satisfied: $\|\boldsymbol{Q}_1\| \gg \|\boldsymbol{Q}_2\|$.
- $\boldsymbol{J}_1 = \begin{bmatrix}\boldsymbol{J}_c & \boldsymbol{I}_{3\times 3} & \boldsymbol{0}_{3\times 6} & \boldsymbol{0}_{3\times 6} & \boldsymbol{0}_{3\times 12} \\ \boldsymbol{J}_{lf} & \boldsymbol{0}_{6\times 3} & \boldsymbol{I}_{6\times 6} & \boldsymbol{0}_{6\times 6} & \boldsymbol{0}_{6\times 12} \\ \boldsymbol{J}_{rf} & \boldsymbol{0}_{6\times 3} & \boldsymbol{0}_{6\times 6} & \boldsymbol{I}_{6\times 6} & \boldsymbol{0}_{6\times 12}\end{bmatrix}$, $I_{n\times n}$ is the identity matrix.
- $\boldsymbol{J}_2 = \begin{bmatrix}\boldsymbol{J}_{lh} & \boldsymbol{0}_{6\times 15} & \boldsymbol{I}_{6\times 6} & \boldsymbol{0}_{6\times 6} \\ \boldsymbol{J}_{rh} & \boldsymbol{0}_{6\times 15} & \boldsymbol{0}_{6\times 6} & \boldsymbol{I}_{6\times 6}\end{bmatrix}$
- $\dot{\boldsymbol{r}}_1 = \begin{bmatrix}\dot{\boldsymbol{r}}_c \\ \dot{\boldsymbol{r}}_{lf} \\ \dot{\boldsymbol{r}}_{rf}\end{bmatrix}$ and $\dot{\boldsymbol{r}}_2 = \begin{bmatrix}\dot{\boldsymbol{r}}_{lh} \\ \dot{\boldsymbol{r}}_{rh}\end{bmatrix}$
- $\boldsymbol{X}^+ = \begin{bmatrix}\tilde{\dot{\boldsymbol{q}}}^+ \\ \boldsymbol{\delta}_1^+ \\ \boldsymbol{\delta}_2^+\end{bmatrix}$ and $\boldsymbol{X}^- = \begin{bmatrix}\tilde{\dot{\boldsymbol{q}}}^- \\ \boldsymbol{\delta}_1^- \\ \boldsymbol{\delta}_2^-\end{bmatrix}$

The QP problem (6) can be solved in real-time using an appropriate QP solver such as uQuadProg solver [11] or qpOASES solver [12].

C. Mapping, localization and replanning

To move in a cluttered environment, a robust and precise sensing input is primordial to determine the position of obstacles, detect collisions and to plan valid long term and short term paths. Also, odometry drift must be constantly verified and corrected by a localization mechanism to ensure a close monitoring of the planned path. The human-size humanoid robots, such as HRP-2 or the humanoids robots which participated in DARPA Challenge, are able to build a 3D map on the fly using their very sophisticated proprioceptive and exteroceptive sensors. However, the Nao robot has only two cameras in the head for sensing and localization. A first approach would be using those cameras, however this has proven to be a very difficult task [13] [14]. Indeed, as explained previously, a humanoid robot swings laterally while walking, which leads to pictures of poor quality. Furthermore, the field of view of the Nao is greatly obstructed by the large table and load. As a result, it is hard to precisely determine the position of the environment and obstacles with respect to the robot.

A second approach would be adding a Kinect camera on the top of Nao's head for mapping [15], in our case a second Kinect camera placed on the front of the cart would probably improve the quality of the 3D mapping. This approach will be studied in a future work.

Our main purpose in this paper is to validate the motion planning approach and whole body control scheme, we therefore, as many related research [16], [17], [18], only to cite few, opted for a complete external sensing and localization. Our system is a Vicon motion capture system constituted of 8 MX20 and 4 T40 cameras. It runs at 100 Hz with a precision of 1 mm.

Markers are placed on the obstacles to construct the 2D costmap of the environment. A global costmap keep tracks of the initial obstacles positions for long term planning, while the local costmap is updated every time a marker moves to ensure safe short term planning and real-time obstacle tracking.

Four markers are placed on a trapezoid shape on the top of Nao's head, they are used to keep track of the robot position and orientation. Another four markers are placed on each corners of the cart and they are also used to track the cart position and orientation.

A collision might occur if an obstacle has been moved or a drift from the planned trajectory happened. When a collision is foreseen, a replanning is necessary as shown in Fig. 4. The new collision-free trajectory is found by the algorithm ARA* starting from the point at which the collision has been predicted. If the potential collision is due to drift, the Dijkstra grid does not need to be recalculated, accelerating therefore the replanning. As the walking pattern trajectory for a humanoid robot cannot be changed instantly, a time interval t_c is required to change the planned footprints. In

the implementation of ZMP preview control, a finite time horizon of 2 steps is used to compute the CoM trajectory. Therefore, if a collision is foreseen at instant t_c, the new collision-free trajectory provided by the algorithm ARA* is deformed to keep the next two footprints unchanged as shown in Fig. 4, the robot will however stop if the deformed trajectory is in collision.

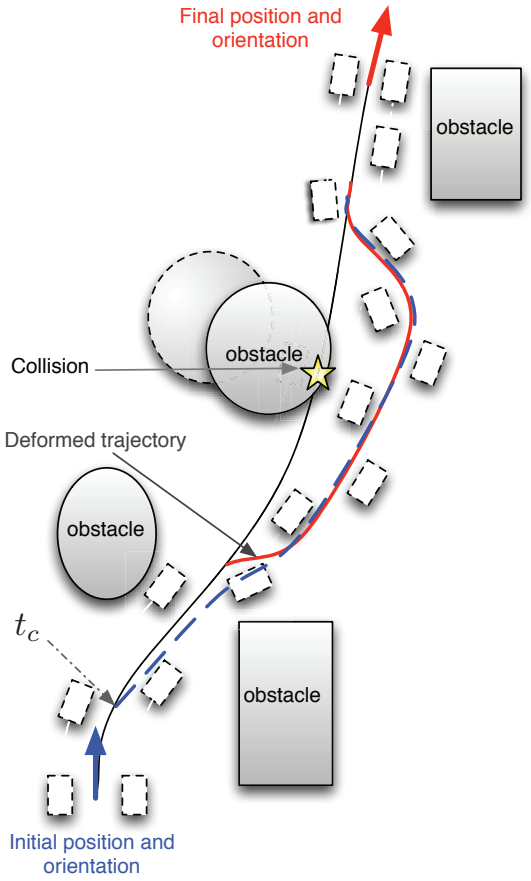


Fig. 4. Replanning in case of collision detection: t_c is the instant at which a collision is foreseen, the new collision-free trajectory is in dashed-blue line, the deformed trajectory is in red line.

IV. Results

Experiments were conducted on a Nao humanoid robot (Fig. 1), manufactured by Aldebaran Robotics [19]. On top of its head, we added 4 motion capture markers in order to track the robot position and orientation.

The cart-like object, shown in Fig. 1, is a mini table 600mm long by 300mm wide. On one side, the two legs are 300mm high and are set on omnidirectional wheels. On the other side, the two legs are half the length so that the Nao can fully support this side of the table. Furthermore, these legs are round and small to fit more tightly in Nao's hands.

The primary objective of our approach is to increase the maximum load weight carried by a Nao humanoid robot, without destabilizing it, while maintaining sufficient flexibility and agility. For that purpose, an experiment aimed at measuring the maximum carrying capacity of the Nao without any modifications has been carried out. Nao is placed in the same pose as in Fig. 1, then a small board (333g) is attached to both hands and is used to support various amount of calibration weights. Even with low weight, the robot is rapidly out of balance and has difficulty following planned trajectories, this is because stabilization algorithms are constantly prioritized to avoid falling. At 300g additional weight, however, Nao fell within the first steps nearly every run, which we determined to be its maximum carrying limit.

With the introduction of the cart, two sets of motion primitives are available, an omnidirectional and a heavy load sets. An example of a right turn for both set is shown in Fig. 2. The carry load at which the friction becomes too high to consider the heavy load set is 700g. In reality, the robot can push higher load, however the wood structure of the cart-like objet cannot support a load higher than 7,000g. To summarize, the maximum carrying capacity of the Nao robot alone is 633g and by using the cart, it is 7,000g, which is 11 times its normal capacity.

In the case of omnidirectional set, the hands and arms are strong enough to articulate the cart while turning to obtain smooth trajectories. The maximum angle at which our robot can turn the cart is 30 degrees, as illustrated in Fig. 5. Over that limit, one hand is colliding with the torso while the other lies outside of the robot workspace.

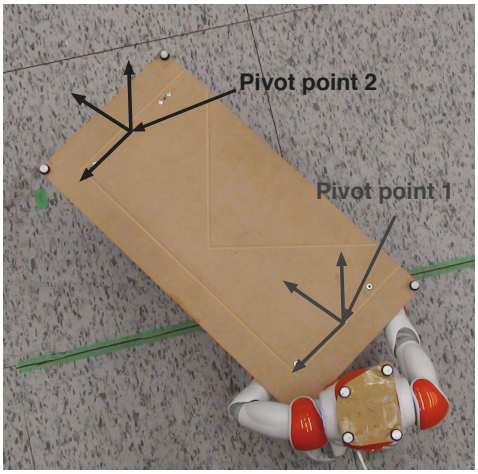


Fig. 5. The arms posture while turning the cart at maximum angle (30 degrees)

While walking in a straight line of 1 m, our Nao is affected by a drifting of an average of 10.5 cm. Without any corrections, this error would lead the robot to constantly diverge from the planned trajectories. The correct localization information provided by the external sensing allow us to modify the trajectory to cancel the drift .

While pushing heavy load, however, the robot cannot rotate in place or move laterally to cancel any drift errors, a quick replanning is therefore executed when the robot diverges too much from the planned trajectory.

As showed in Fig. 6, without any hand position correction, the lateral swing causes large oscillations that are transmitted

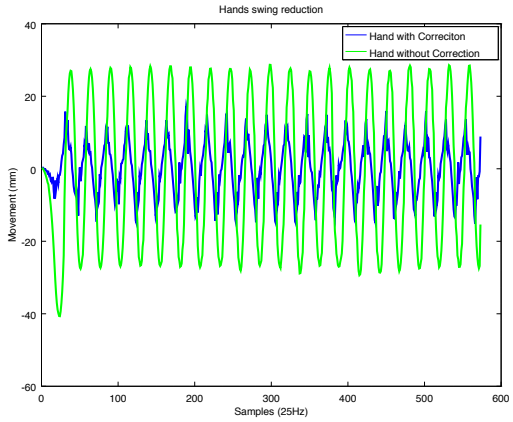


Fig. 6. The hands lateral movement while walking with and without corrections

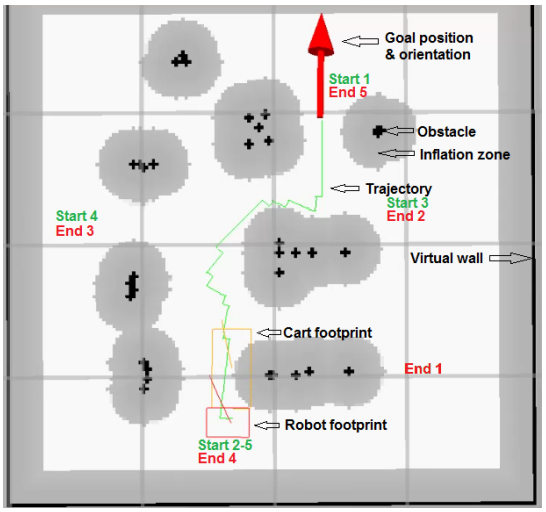


Fig. 7. Map of the obstacles (black), the inflation zone around them (grey) and experiments with the Nao (red square), the cart (orange rectangle) and goal position/orientation (red arrow). Note that the initial posture collides with the inflation zone, but it is not in collision with the obstacles.

to the table and the load. The average peak-to-peak position movement is 56.25 mm. However, with the proposed corrections, the hand distance from desired position has been reduced to 28.20 mm, reducing the hand error by 49.87%. As a result, significantly less oscillations are transmitted to the table, leading to a safer and enhanced carrying ability and load stability. However, note that the error cannot be completely cancelled because: I) the hand trajectories are second priority task, that means the robot will respect those trajectories as far as the trajectories of first priority are fully followed, II) the Nao robot has only 5 degrees of freedom in each arm, III) the joint backlash of the Nao robot.

To test the system as a whole and to validate the proposed algorithms, we conducted three series of 5 experiments. The omnidirectionnal set of primitive has 12 different primitives with θ_r sampling of $11.25°$ while the heavy set have only 5 possible primitives, but with a precision of $5.625°$. In each experiment, the robot starting and goal positions were chosen in a way that the robot had to navigate through a field of motion capture markers on the ground. Every marker serves as an obstacle that must be avoided by the robot and the cart. They were placed to form various feasible paths and force tight turns in order to take advantage of the additional degree of freedom (the rotation of the cart θ_{ca}). The three series were composed of the same experiments containing the same initial configuration of the obstacles in the environment and using the same initial and goal positions and orientations, but with different transported objects.

Fig. 7 shows the start and end positions for each type of experiment. In this figure, the obstacles are in black, while the grey areas around them are an inflation zone where the cost is higher than free space to prevent the robot from passing too close to obstacles. The vertical and horizontal black lines are virtual walls to prevent the planner from detecting optimal trajectory passing around the experimental setup.

The first series consists of the Nao robot alone, without a cart or load. It uses the omnidirectional primitives set to navigate through the obstacles. The second one was conducted with the Nao holding the cart, which increased the navigation footprints significantly. These tests also used the omnidirectional set to construct the plan. For the third and final series, the robot is pushing the cart with an additional load of 2,300g, in this case the heavy load set of primitives have been used to allow the Nao to plan a trajectory in the cluttered environment.

The ARA* planner initial $\epsilon = 3$ means that the suboptimal solution cannot be worse than 3 times the cost of the optimal solution. A time limit of 10 seconds was chosen and within that time ϵ successfully decreased to 1 on every run, which corresponds to the optimal solution. For each generated path, we measured the total time to execute the trajectory, the trajectory length, the initial solution time, the optimal solution time, the initial expanded nodes and the final expanded nodes. The PC that has been used to generate these results has: a i7-3770 Processor with 8 cores at 3.4GHz and 8GB of RAM. These results are summarized in Table II.

	No Cart	Cart Omni	Cart Heavy Load
Total time (s)	69.75	104.90	134.71
Total time Std (s)	14.69	15.76	27.62
Trajectory length (m)	2.41	3.03	3.20
Trajectory length Std (m)	0.53	0.26	0.48
Average velocity (m/s)	0.035	0.029	0.024
Initial solution time ($\epsilon = 3$) (s)	0.020	0.026	0.012
Initial solution time Std (s)	0.020	0.024	0.008
Optimal solution time ($\epsilon = 1$) (s)	0.20	0.41	0.44
Optimal solution time Std (s)	0.11	0.46	0.38
Initial node expansions	541.6	769.8	892.4
Initial node expansions Std	492.5	497.0	508.7
Total node expansions	4775	10694	25165
Total node expansions Std	2760	11436	24105

TABLE I
EXPERIMENTAL STATISTICS

It can be observed that the average velocity is lower while using the cart. This is explained by the friction of the wheels of the cart causing slippage as the robot tries to move,

slowing its movements down. Every step in a direction results in a slippage in the opposite direction, thus progressing less distance with each step. This leads to a reduced speed of 17.14% for when pushing the empty cart and 31.43% speed reduction with the additional 2.3 kg load.

Even-though the primitives with the robot alone and with the cart are the same, it is hard to find a path as optimal with it. Since the length is only about 25% longer though, moving with the table does not impair too much the robot ability to travel the cluttered environment swiftly.

The weight primitives trajectory length however is higher in comparison, about 33% higher than the Nao alone. This is primarily due to the change of primitives. The movement is not as smooth, in particular, moving sideways or in diagonal become impossible. Also, turning in place versus turning around a pivot placed $r = 0.60m$ away increased the trajectory by at least $L = \frac{\theta \pi r}{180}$ every time the robot makes a turn.

The biggest problem we encountered is the very limited space where the Vicon is precise. Indeed, the precision is under 1 mm while correctly positioned, but during our experimentations, often when Nao got too close to the edges of the "sweet spot", it starts to lose the markers and fails to follow correctly and consistently the localization. As a result the working space was limited, however our algorithm is more generic and already ready for large scale autonomous navigation. We plan changing this system for a more autonomous one as it will be explained in Section V.

Also, the friction of the wheels to the ground is not high enough to completely prevent them from sliding and thus to act as a perfect pivot while using heavy load. This causes additional errors when turning and walking that can accumulate and force a replanning. For this reason, with the heavy load, 1.8 replanning were needed on average, while only 0.4 for the cart alone and 0 without the cart when no obstacles are moved.

When an obstacle is moved in the initial trajectory of the robot or cart, a replanning is essential to avoid collision. On all our experiments, there was no collision between neither the Nao or the cart with any obstacles or the virtual walls[1].

V. Conclusion and Future Work

In future work, our priority will be to remove any dependency on the external localization system, that is the Vicon system, and instead use other techniques to make the system completely autonomous. As suggested in Section III-B, adding Kinects to the Nao and the table will be explored. Also, since we are using multiple sets of primitives depending on the load weight, an automatic estimation of the load weight, to decide which set to be used, would improve the system and make it more general and autonomous.

Acknowledgment

We thank the anonymous reviewers for their helpful comments. This research is supported by a discovery grant (Prof. Wael Suleiman) from the Natural Sciences and Engineering Research Council of Canada (NSERC).

[1] A video of the experiments is available at https://goo.gl/RYUAY7

References

[1] C Ronald Kube and Eric Bonabeau. Cooperative transport by ants and robots. *Robotics and autonomous systems*, 30(1):85–101, 2000.

[2] ZhiDong Wang, Majid Nili Admadabadi, Eiji Nakano, and Takayuki Takahashi. A multiple robot system for cooperative object transportation with various requirements on task performing. In *IEEE International Conference on Robotics and Automation, 1999*, volume 2, pages 1226–1233.

[3] Atsushi Yamashita, Tamio Arai, Jun Ota, and Hajime Asama. Motion planning of multiple mobile robots for cooperative manipulation and transportation. *IEEE Transactions on Robotics and Automation, 2003*, 19(2):223–237.

[4] Satoshi Shigemi, Yuichiro Kawaguchi, Takahide Yoshiike, Koji Kawabe, and Naohide Ogawa. Development of new ASIMO. *Honda R and D Technical Review, 2006*, 18(1), 2006.

[5] Shunichi Nozawa, Toshiaki Maki, Mitsuharu Kojima, Shigeru Kanzaki, Kei Okada, and Masayuki Inaba. Wheelchair support by a humanoid through integrating environment recognition, whole-body control and human-interface behind the user. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008*, pages 1558–1563.

[6] Michael Stilman and James Kuffner. Navigation among movable obstacles: Real-time reasoning in complex environments. In *Proceedings of the 2004 IEEE International Conference on Humanoid Robotics (Humanoids'04)*, volume 1, pages 322 – 341, December 2004.

[7] Jonathan Scholz, Sachin Chitta, Bhaskara Marthi, and Maxim Likhachev. Cart pushing with a mobile manipulation system: Towards navigation with moveable objects. In *IEEE International Conference on Robotics and Automation, 2011*, pages 6115–6120.

[8] Maxim Likhachev, Geoffrey J Gordon, and Sebastian Thrun. ARA*: Anytime A* with provable bounds on sub-optimality. In *Advances in Neural Information Processing Systems*, page None, 2003.

[9] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. Biped Walking Pattern Generation by using Preview Control of Zero-Moment Point. In *Proc. IEEE International Conference on Robotics and Automation*, pages 1620–1626, Taipei, Taiwan, 2003.

[10] F. Kanehiro, F. Lamiraux, O. Kanoun, E. Yoshida, and J.-P. Laumond. A Local Collision Avoidance Method for Non-strictly Convex Objects. In *2008 Robotics: Science and Systems Conference*, Zurich, Switzerland, June 2008.

[11] Fumio Kanehiro, Mitsuharu Morisawa, Wael Suleiman, Kenji Kaneko, and Eiichi Yoshida. Integrating geometric constraints into reactive leg motion generation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4069–4076, 2010.

[12] H.J. Ferreau, C. Kirches, A. Potschka, H.G. Bock, and M. Diehl. qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 2014. (in print).

[13] Olivier Stasse, Andrew J Davison, Ramzi Sellaouti, and Kazuhito Yokoi. Real-time 3d slam for humanoid robot considering pattern generator information. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006*, pages 348–355.

[14] Stefan Oßwald, Armin Hornung, and Maren Bennewitz. Learning reliable and efficient navigation with a humanoid. In *IEEE International Conference on Robotics and Automation, 2010*, pages 2375–2380.

[15] D. Maier, A Hornung, and M. Bennewitz. Real-time navigation in 3d environments based on depth camera data. In *12th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 692–697, Nov 2012.

[16] S. Hak, N. Mansard, O. Stasse, and J.P. Laumond. Reverse control for humanoid robot task recognition. *IEEE Transactions on Systems, Man, and Cybernetics–Part B : Cybernetics*, 42, 2012.

[17] M. Levihn, K. Nishiwaki, S. Kagami, and M. Stilman. Autonomous environment manipulation to assist humanoid locomotion. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4633–4638, May 2014.

[18] Seungsu Kim, A. Shukla, and A. Billard. Catching objects in flight. *IEEE Transactions on Robotics*, 30(5):1049–1065, Oct 2014.

[19] David Gouaillier and *et al*. Mechatronic design of NAO humanoid. In *IEEE International Conference on Robotics and Automation, 2009*, pages 769–774.

2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)
November 3 - 5, 2015, Seoul, Korea

Cooperative SLAM-Based Object Transportation by Two Humanoid Robots in a Cluttered Environment

Antoine Rioux, Claudia Esteves, Jean-Bernard Hayet and Wael Suleiman

***Abstract*— In this work, we tackle the problem of making two humanoid robots navigate in a cluttered environment while transporting a very large object that simply can not be moved by a single robot. We present a complete navigation scheme, from the incremental construction of a map of the environment and the computation of collision-free trajectories to the control to execute those trajectories. We present experiments conducted on real Nao robots, equipped with RGB-D sensors mounted on their heads, moving an object around obstacles. Our experiments show that a significantly large object can be transported without changing the robot's main hardware, and therefore enacting the capacity of humanoid robots in real-life situations.**

I. Introduction

One of the advantages of having arms on a robot is that it can manipulate a load. This capacity can be useful for a wide range of actions, including transporting objects from one place to another. This can be particularly useful for automated construction and rescue missions situations. However, using one robot only, the maximum payload is generally low and the size of transported objects limited. One way to deal with this issue is to distribute the weight to multiple robots. In this work we deal with the problem of having two humanoid robots cooperating to handle a bulky object among obstacles.

This problem has been tackled in previous works using mainly two approaches: (1) a leader-follower control, or (2) a synchronized control. In the first approach [1] [2], one of the robots, the leader, based on its position and its surrounding, computes the plan for the system or is directly guided by a human operator. The second robot follows the leader. This technique is easy to implement, but does not allow closed-loop cooperation easily, because, as the follower only responds to the leader movement when it has already started, a significant time delay is introduced.

Most human-robot cooperative system use this approach [3] [4] since these problems are alleviate by adding a human into the loop, which instinctively correct both micro and macro errors in the system.

In the second approach [5] [6], an external centralized controller computes the motion for all the robots simultaneously, based on the information of the environment provided by the robots. As a result, the synchronized motions can start and stop together. However, both synchronization system rely on "step synchronization" instead of position synchronization. While easier to synchronized, this process slows down the system significantly and increased lateral instability as the entire system swing from side to side in harmony. Furthermore, the transported object is held rigidly and cannot be articulated to improve the footprint and motions of the system. In this work, which belongs to the latter approach, a framework for synchronized cooperative autonomous humanoid robots navigating in a cluttered environment while manipulating an object in closed-loop is presented.

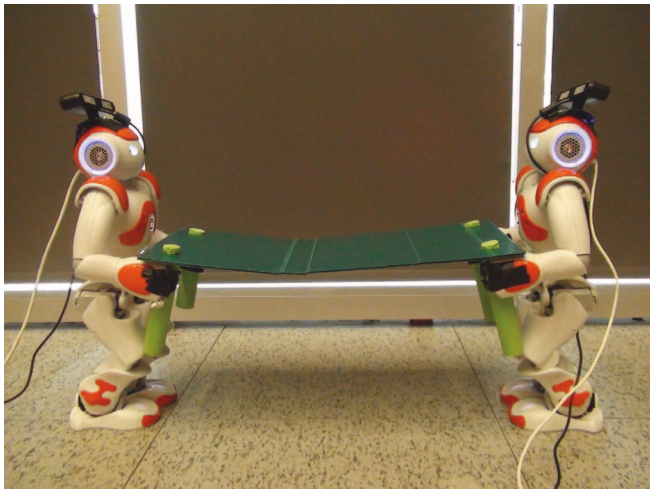


Fig. 1. The Nao robots holding an object together

Our main contributions are: (1) a low dimensional multi-robot motion planning algorithm to find an obstacle-free trajectory using a map of the environment autonomously constructed by the robots, (2) a continuous and consistent odometry system integrating the robots visual data and actuators information, (3) a synchronization strategy that uses the projection of the robots, and (4) an efficient real-time whole-body control scheme that generates the motions of the closed-loop robot-object-robot system. The remainder of the paper is organized as follows: Section II presents the proposed planning algorithm. In Section III, a brief description of how the map of the environment is constructed is given. Section IV details the proposed synchronization approach and Section V describes the control scheme to execute the planned trajectories. Finally, in Section VI, results of simulation and real world experiments are presented and discussed.

A. Rioux and W. Suleiman – Electrical and Computer Engineering Department, Faculty of Engineering, University of Sherbrooke, Canada; C. Esteves – Department of Mathematics. Universidad de Guanajuato, México; J-B. Hayet – Centro de Investigación en Matemáticas (CIMAT), Guanajuato, México. {antoine.rioux, wael.suleiman}@usherbrooke.ca, {cesteves, jbhayet}@cimat.mx

978-1-4799-6885-5/15/$31.00 ©2015 IEEE

II. Planning a valid path

To navigate through a cluttered environment, the computation of a collision-free path for both robots is essential. For this, we chose a lattice-based graph planning with an ARA* search [7], motivated by the use of motion primitives ensuring feasible robot-object-robot configurations and transitions. The environment is modelled by a 2D grid cost-map that discriminates obstacles from free space at a fixed threshold and allows obstacles inflation to increase the security margin.

A. State representation

Each node of the search graph needs a representation of the complete robots-object state. To achieve this, it is possible to model the state in $\mathbb{R}^2 \times \mathbb{S}^1 \times \mathbb{R}^2 \times \mathbb{S}^1 \times \mathbb{R}^2 \times \mathbb{S}^1$:

$$s = (x_{r1}, y_{r1}, \theta_{r1}, x_{ob}, y_{ob}, \theta_{ob}, x_{r2}, y_{r2}, \theta_{r2}), \qquad (1)$$

where x_{ri}, y_{ri} and θ_{ri} $(i = 1, 2)$ are the positions and orientation of the i-th robot, and x_{ob}, y_{ob} and θ_{ob} are those of the object. As the working space of our robot's arms is too small to fully take advantage of both rotation and translation, the system is simplified by setting a pivot point at the middle of the pair of hands for both robots shown in Fig. 2. The closed-loop grasping of the robot on the table is shown in Fig. 1. The pivot points positions maximize the rotation range within the robot workspace, resulting in a smaller 5 dimensions state space $\mathbb{R}^2 \times \mathbb{S}^1 \times \mathbb{S}^1 \times \mathbb{S}^1$:

$$s = (x_{r1}, y_{r1}, \theta_{r1}, \theta_{ob}, \theta_{r2}). \qquad (2)$$

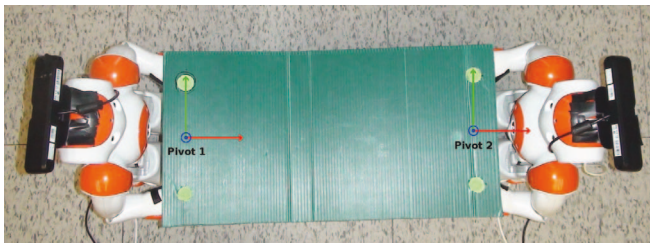


Fig. 2. Top view: Pivots position

Even though the above simplification removes the ability of the object to translate on the plane, the robot retains enough manipulability to minimize the robot-object-robot collision area around obstacles. The simplified state representation of equation (2) is shown in Fig. 3.

In a lattice-based graph planner, transitions between nodes are triggered by actions chosen within a finite fixed-set of motion primitives. An important feature of the lattice representation is that all connections are feasible paths. Therefore, it is really suitable for highly constrained systems, such as a system of two robots transporting an object, in contrast to other commonly used forms of graph search, including Von Neumann or Moore neighborhood.

The set of motion primitives used for this problem is showed in Fig. 4. It includes (a) forward, backward, sideway motion and every diagonal motion, (b) rotations around each robot and the object center and special movements such as

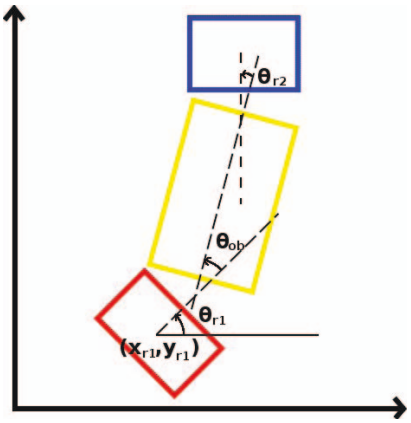
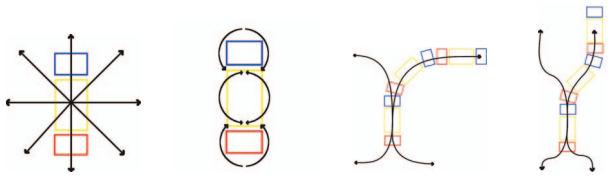


Fig. 3. Simplified state representation.



(a) Linear (b) Rotation (c) C-turn (d) S-turn

Fig. 4. Set of possible motion primitives.

(c) C-turns and (d) S-turns. The last two are more complex and make use of the hands articulations to increase agility around obstacles. Executing complex motions specific to a system in a coherent and logical way is the main reason we use motion primitives over homogeneous sampling of the system DOFs. The joints of the system that allow these configurations are the aforementioned pivot points.

B. Path Cost Function

The cost of a transition from state s to s' is based on the time to execute that transition and is computed as follows:

$$g(s, s') = \begin{cases} \frac{\sqrt{(\Delta x_{r1})^2 + (\Delta y_{r1})^2}}{\dot{r_1}^+} \times DF & \text{if } \Delta x_{r1} \neq 0 \text{ or } \Delta y_{r1} \neq 0 \\ \frac{\sqrt{(\Delta x_{r2})^2 + (\Delta y_{r2})^2}}{\dot{r_2}^+} \times DF & \text{otherwise} \end{cases} \qquad (3)$$

where Δx_{ri}, Δy_{ri} are the variations of the x and y coordinates of the i-th robot pelvis, between states s and s', DF is a difficulty factor associated with each primitive, $\dot{\boldsymbol{r}}^+$ is the robot maximal linear velocity. This ratio gives us the approximate time to execute the primitive. Since the state representation does not contain any information about the position of the second robot, we must determine Δx_{r2} and Δy_{r2} with the first robot position. To do so, we compute the projection of the object by rotating the first robot by θ_{r1}, given a hand rotation of θ_{ob}. Then, we project the second robot around the object given a hand rotation or θ_{r2}.

The difficulty factor DF is a special value set by the system expert in order to prioritize or penalize certain motions. For example, turning in place then going forward takes longer to execute than moving in diagonal. For relatively long

distance, however, moving in a straight line has a smaller trajectory footprint, is more natural looking, minimizes walking oscillation and causes less drift and slippage than moving in diagonal. For those reasons, all the diagonal primitives have higher DF than turning in place and moving forward. The difference in cost will still, however, favor a diagonal movement where turning in place is not worth it, i.e. for short diagonal movements.

C. Search Algorithm

A* is one of the most popular search algorithms. In addition to the use of a path cost function, a heuristic biases the search towards the most promising states. Even though A* is optimal when it finds a solution, that solution does not always exist or cannot be found within a reasonable time. The Anytime Repairing A* (ARA*) focuses on delivering a suboptimal solution as fast as possible; this solution is then optimized iteratively within a predefined limited time. Also, the states are expanded from goal to start, so that the heuristic costs remain valid after replanning and do not need to be recomputed. The cost function takes the form of:

$$f(s,s') = g(s,s') * max(Cost_{cells}(s,s')) + \epsilon h, \epsilon >= 1 \quad (4)$$

where $g(s,s')$ is the path cost of equation (3), h is the heuristic that uses a 2D grid containing all the Dijkstra distance costs from the goal to the start states and

$$Cost_{cells}(s,s') = \begin{cases} 1 & \text{free space} \\ 2 \text{ to } 99 & \text{inflation} \\ \infty & \text{obstacles} \end{cases} \quad (5)$$

includes the cost of cells between s and s'. The search is biased towards states closer to goal and returns a solution that is, at worst, ϵ times the cost of the optimal solution.

The inflation is a zone around obstacles where all the cells have a higher cost. It is used as a security margin to bias the search farther from obstacles and reduce danger of collisions. It can be set as a decreasing gradient from the obstacles or a fixed value in the range above.

III. Simultaneous Localization and Mapping

To navigate in a cluttered environment, robust and precise sensing is primordial to determine the position of obstacles, detect collisions and to plan valid paths. Also, odometry drift must be constantly corrected by an accurate localization to ensure that planned paths are closely followed. However, the Nao robot has only two cameras in its head for sensing and using them for localization has proven to be a very difficult task [8], because of the robot sway motion and the low resolution pictures. Furthermore, in our case, the field of view of the Nao is greatly obstructed by the object being transported and by the other robot.

A. Real-Time Appearance-Based Mapping (RTAB-Map)

We chose to add a depth camera on the top of each Nao's head for mapping [9], based on RTAB-Map [10], [11]. RTAB-Map provides a robust odometry system based on visual information. It can also create 3D maps of the environment as well as constructing a 2D occupancy grid map by projecting the obstacles on the ground plane.

B. Odometry fusion

A problem that occurs when using the visual odometry produced by RTAB-Map is that it may lose track of the position for multiple reasons, such as lack of detected features in the observed environment, rapid movements of the camera or intense oscillations. When this occurs, we could go back to where the tracking was lost, but this is not efficient and may even be impossible. For this reason, a fusion of the visual odometry, the robot's internal odometry and the error between those reference frames is used to improved the overall odometry.

Since the camera is rigidly linked to the robot by a transformation T_a^v, we can write $T_a^o = T_a^v * T_v^o$, where T_a^o, T_v^o are the homogeneous transformation matrix between the map frame and, respectively, the robot frame and the camera frame. The previous equation can be rewritten to include the encoders-based robot odometry T_r^o, that does not take into account slipping, drift and other real world errors,

$$\begin{aligned} T_a^o &= T_a^v * T_v^o * T_r^{o-1} * T_r^o \\ &= T_a^v * T_v^r * T_r^o, \end{aligned} \quad (6)$$

where T_r^v is the error between the encoders odometry and the visual odometry. Since this equation only holds while the visual odometry is valid, the last valid T_v^r at time $t = t_{lost}$ is used when a loss occurs at $t = t_{lost}$,

$$T_a^o(t) = \begin{cases} T_a^v * T_v^r(t) * T_r^o(t) & \text{if } T_v^o \text{ exists,} \\ T_a^v * T_v^r(t_{lost}) * T_r^o(t) & \text{otherwise.} \end{cases} \quad (7)$$

This approach consistently provides smooth odometry. Even when the visual information is abruptly discontinued, it continues to generate sufficiently accurate localization data until an adequate image or a reset command is processed by RTAB-Map and the visual odometry is restored.

IV. Synchronization

A. Object Stability and Hand Stabilization

At low speed, a humanoid robot CoM moves horizontally from one support foot to the other in order to stay in balance. This lateral motion causes the entire upper body to oscillate laterally at an amplitude proportional to the distance between the center of its feet, which, in our case, causes the transported object to move by the same amplitude.

Since the object to carry is fully controlled by the robots' hands, it is possible to reduce this effect so as to improve the closed-loop stability. On the one hand, if both robots swing at the same time, synchronization is maintained without effort, but the object and anything on it would swing dangerously. On the other hand, if the robots swing in any other way, the force generated by this movement will be transmitted to the other robot and cause instability or fall.

Our solution to compensate this instability without changing the walking gait is to use the robots' hands and keep them at a fixed position in space, relatively to the planned

trajectory. This position is determined at the starting position of the robot and corresponds to the transformation between feet and hands. Those transformations will be the input of the whole-body control scheme described in Section V.

B. Synchronized trajectories

Even if both robots are independent and receive their own trajectory to follow, these trajectories are lined up in such a way that they are at a constant distance from each other. However, real robots being imperfect, do not necessarily move at the exact same speed given the same command. For this reason, the trajectory is segmented into waypoints, each waypoint being a state of the graph plan. To progress to the next waypoint, both robots have to agree that they are close enough to their current waypoint in term of position and orientation. If only one robot is near its waypoint, it will slow down while converging to it, until the other robot agrees that they can proceed to the next waypoint.

C. Synchronized projections

Now that the whole system is more stable with regards to the individual swing added by both robots, the position of each robot needs to be synchronized with the other one along the planned trajectory. To do so, the projection of each robot with respect to the other is computed by using their respective hands, world frames and transported object properties. First, the position of the center of the object with respect to a robot i can be found by:

$$T_{ob}^{r_i} = midpoint(T_{h_r}^{r_i} * T_{ob}^{h_r}, T_{h_l}^{r_i} * T_{ob}^{h_l}) \tag{8}$$

where $T_{ob}^{r_i}$, $T_{h_{(r,l)}}^{r_i}$, $T_{ob}^{h_{(r,l)}}$ are respectively the current transformations between the robot and the object frames, the robot and its hands and the object center. The function $midpoint$ computes the midpoint of two points.

With these transformations, we find the projection synchronization position for each robot as follows:

$$T_{p_i}^{o} = T_{r_j}^{o} * T_{ob}^{r_j} * T_{ob}^{r_i\,-1} \qquad \text{for } i,j \in \{1,2\} : i \neq j, \tag{9}$$

where $T_{p_i}^{o}$ is the projected robot i position in the world frame and $T_{r_i{}^c}^{o}$ is the odometry data from the other robot i^c. Other points different from the center could be used instead, such as the pivot points. However, a constant offset transformation would need to be taken into consideration in the previous equation. We can finally compare this projected position to the actual position of each robot in order to find the projection synchronization error e_{p_i}

$$e_{p_i} = T_{p_i}^{o} * T_{r_i}^{o\,-1}. \tag{10}$$

D. Synchronization as Kinematics tasks

Our objective is to express the synchronization between the two robots as kinematics tasks that can be solved using a whole-body control scheme:

- The hands synchronization can be expressed as a kinematics task on the hands positions of each robot.
- The synchronized projection can be expressed as a kinematics task on the chest frame of each robot. More precisely, this kinematic task is on the horizontal positions (x and y) and the orientation around the vertical axis (z) of the chest frame.

V. Control

Once a collision-free trajectory is found by the ARA* algorithm, a set of footprints are defined along the trajectory as shown in Fig. 5. The second step is to define a Zero Moment Point (ZMP) trajectory. A trajectory of the Center of Mass (CoM) of the robot is then obtained using the preview control algorithm proposed in [12]. This algorithm, widely used in humanoid robotics, is simple to implement, yet efficient and yields a smooth CoM trajectory by minimizing the CoM jerk trajectory. The feet trajectories are obtained by spline interpolation between the footprints and the hands trajectories and orientations are defined in order to minimize the walking swing effect as well as to follow the object orientation.

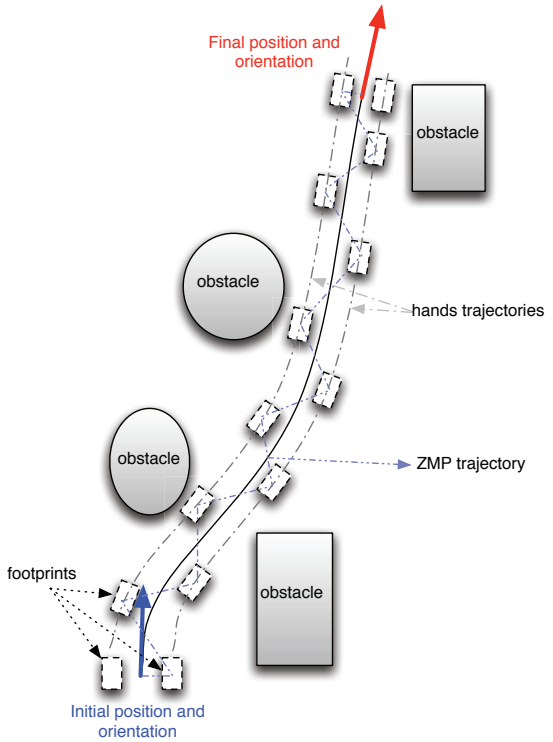


Fig. 5. Overview of the motion planning procedure

To obtain the joint trajectories for each humanoid robot, a whole-body control scheme with prioritized tasks is formulated as follows:

$$\min_{\dot{\boldsymbol{q}}} \dot{\boldsymbol{q}}^T \boldsymbol{Q} \dot{\boldsymbol{q}}$$

subject to

$$\begin{array}{ll} \text{First priority} & \begin{cases} \boldsymbol{J}_c \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_c \\ \boldsymbol{J}_{lf} \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_{lf} \\ \boldsymbol{J}_{rf} \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_{rf} \end{cases} \\ \text{Second priority} & \begin{cases} \boldsymbol{J}_{lh} \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_{lh} \\ \boldsymbol{J}_{rh} \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_{rh} \\ \boldsymbol{J}_{ch} \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_{ch} \end{cases} \\ \text{Joint velocity limits} & \dot{\boldsymbol{q}}^- \leq \dot{\boldsymbol{q}} \leq \dot{\boldsymbol{q}}^+ \end{array} \tag{11}$$

where $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ is the joint velocity vector, $\boldsymbol{Q}$ is a positive semi-definite matrix, $\boldsymbol{J}_c \in \mathbb{R}^{3\times n}$, $\boldsymbol{J}_{lf} \in \mathbb{R}^{6\times n}$, $\boldsymbol{J}_{rf} \in \mathbb{R}^{6\times n}$, $\boldsymbol{J}_{lh} \in \mathbb{R}^{6\times n}$, $\boldsymbol{J}_{rh} \in \mathbb{R}^{6\times n}$, $\boldsymbol{J}_{ch} \in \mathbb{R}^{3\times n}$ are the Jacobian matrices of CoM, left foot, right foot, left hand, right hand and chest, respectively. $\dot{\boldsymbol{r}}_c$, $\dot{\boldsymbol{r}}_{lf}$, $\dot{\boldsymbol{r}}_{rf}$, $\dot{\boldsymbol{r}}_{lh}$, $\dot{\boldsymbol{r}}_{rh}$, $\dot{\boldsymbol{r}}_{ch}$ are the linear and angular velocity of CoM, left foot, right foot, left hand, right hand and chest, respectively. Since we are only interested in the horizontal velocity and angular velocity around the vertical axis for the chest frame, $\dot{\boldsymbol{r}}_{ch} \in \mathbb{R}^3$, and $\dot{\boldsymbol{q}}^-$ and $\dot{\boldsymbol{q}}^+$ are the joint velocity limits.

The optimization problem (11) can be transformed into a standard Quadratic Programming (QP) problem [13], which can be solved in real-time by using an appropriate QP solver.

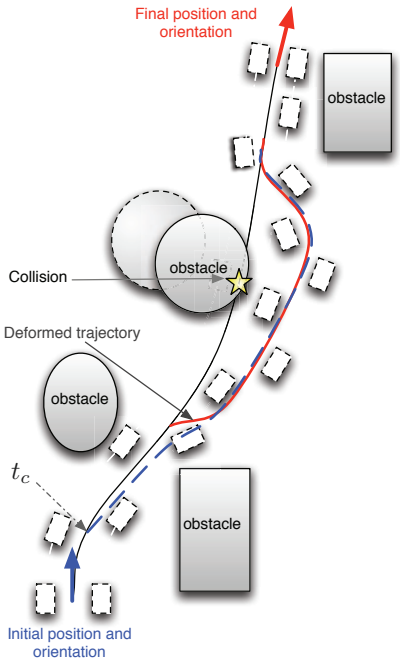


Fig. 6. Replanning in case of collision detection: t_c is the instant at which a collision is foreseen, the new collision-free trajectory is in dashed-blue line, the deformed trajectory is in red line.

A. Dynamic collision avoidance and Replanning

It is important to check frequently for collision in case an obstacle has moved from its original position, the robots drift away from their planned trajectory or the synchronization process of the robots requires a replanning of the whole trajectory. Hence, the future trajectory is always monitored for potential collisions with obstacles in the 2D occupancy grid. When a collision is foreseen, a replanning is necessary and the trajectory is deformed as shown, for the purpose of clarity for a single robot, in Fig. 6. Even though, at first glance, the support polygon seems increased by including the robot-object-robot closed loop, the robots' support polygons are still defined by the contact between the feet and the ground. This is because the robots' arms are not fully bended, therefore the robots could fall forwards or backwards.

The new collision-free trajectory is found by the ARA* algorithm from the goal to the point at which the collision has been predicted. If the potential collision is due to drift and the environment has not changed, the Dijkstra grid does not need to be recalculated, therefore greatly accelerating the replanning. As the humanoid robot walking pattern cannot be changed instantly, a time interval t_c is required to change the planned footprints. In the implementation of ZMP preview control, a finite time horizon of 2 steps is used to compute the CoM trajectory. Therefore, if a collision is foreseen at instant t_c, the new collision-free trajectory provided by the ARA* algorithm is deformed to keep the next two footprints unchanged as shown in Fig. 6. The robot will however stop if the deformed trajectory is in collision. Contrary to what Fig. 6 might suggest, 2 steps do not represent a significant distance. Since the robots are very small and the feet tend to slide, 2 steps are only a few centimetres or less.

VI. Results

Experiments were conducted on Nao humanoid robots (Fig. 1), manufactured by Aldebaran Robotics [14]. Their dimensions are 573mm of height, 311mm of width and 275mm of depth for a total weight of 5.2kg. The two arms as well as both legs have 5 DOF each, while the head has 2 DOF and the pelvis and hands have 1 DOF each. An IMU provides odometry data and 36 magnetic rotary encoders give joint angle information with a precision of 0.1°. On top of their heads, we have added an Asus Xtion Pro Live consumer-level depth camera (see Fig. 1).

A. Articulating the arms

Without any correction, the average oscillation peak-to-peak position movement of the hands is 47.6 mm. However, with the whole body control, the average hand distance from desired position has been reduced to 16.4 mm, reducing the hand error by 65.5%. As a result, significantly less oscillations are transmitted to the table, leading to a safer and enhanced carrying ability and load stability. However, the error cannot be completely cancelled, this is mainly because: I) the hands trajectories are second priority tasks, II) Nao has only 4 DOFs in each arm.

B. Navigating in a cluttered environment

To test the system as a whole and to validate the proposed algorithms, we conducted two series of 5 experiments. In each experiment, the robots starting and goal positions are chosen in such a way that the robots have to navigate

among objects on the ground. Each of them serves as an obstacle to be avoided by the robots and the object. They are placed to form various feasible paths and force tight turns in order to take advantage of the additional degrees of freedom (the rotation of the object θ_{ob} and θ_{r_2}). The two series include the same experiments, except for one series that has the active projection correction and the trajectory synchronization while the other has only the latter.

Fig. 7 shows the start and end positions for each type of experiment. In this figure, the obstacles are in yellow, while the red areas around them are inflation zones where the cost is higher than in free space, to prevent the robots from passing too close to obstacles. These zones are used as a security buffer and the center of the robots and the object should avoid, if possible, planning to pass inside it. The cyan zone is a forbidden zone, because if the center of the object or the robots enters it, it means that an edge is in collision with an obstacle.

The ARA* planner parameter initial value $\epsilon = 3$ means that the suboptimal solution cannot be worse than 3 times the optimal solution cost. A time limit of 5 seconds was chosen and within that time, ϵ was successfully decreased to 1 on every run, which corresponds to the optimal solution. For each generated path, we measured the total time to execute the trajectory, the trajectory length, the initial solution and optimal solution times. These results are summarized in Table I.

	No Sync	With Sync
Total time (s)	83.38	81.82
Total time Std (s)	9.48	9.00
Trajectory length (m)	2.36	2.28
Trajectory length Std (m)	0.30	0.22
Average velocity (m/s)	0.0283	0.0278
Initial solution time ($\epsilon = 3$) (s)	0.026	0.028
Initial solution time Std (s)	0.036	0.04
Optimal solution time ($\epsilon = 1$) (s)	0.352	0.322
Optimal solution time Std (s)	0.305	0.361

TABLE I
STATISTICS ABOUT THE EXPERIMENTS

We observe that the average speed is nearly the same, which is surprising because it was expected to be significantly slower with the projection synchronization. Indeed, for the robots to properly stay synchronized, they try to match their speed and relative position while following their respective trajectory. This means that the fastest robot slows down to accommodate the other while the slowest tries to speed up. Since the difference in average velocity is only about 1.8%, it could also be explained by other factors, such as different battery level and motors temperature.

The slowest robot was most likely already slowing down the fastest one by not agreeing to pass the waypoints, forcing a slow down. The projection synchronization forced the slow one to move through these waypoints faster, causing the slow down on the fastest to be mostly cancelled by these faster waypoint transitions.

Even though in our case the trajectories are quite small, the use of sub-optimal solutions has proven to be significantly faster. Indeed, for our experimental settings, it is about 12.5 times faster to compute the solution for $\epsilon = 3$ as opposed to the optimal solution. This could be especially useful for real life large scale distances and experiments where quick reactions and planning are necessary.

In Table II, the errors in X, Y and Yaw for both robots were measured at 10 Hz during each experiments.

	Robot 1			Robot 2		
	X	Y	Yaw	X	Y	Yaw
No proj (m)	0.032	0.024	-0.009	0.033	0.0175	0.009
No proj Std (m)	0.0025	0.007	0.013	0.002	0.011	0.013
Projection (m)	0.010	0.003	0.006	0.011	0.007	-0.006
Projection Std (m)	0.001	0.004	0.0125	0.002	0.007	0.0125
Error reduction (%)	68.75	87.50	33.33	66.67	60.00	33.33

TABLE II
ERRORS STATISTICS

The error is significantly reduced on every axes for both robots. More importantly, the errors in Y and Yaw converge to zero. However, we still have some error in the X axis, because it is the main direction of movement and the system wants to follow the planned trajectories as a main priority. The focus here is to reduce the errors, but not necessarily cancel them completely if it means that it will prevent normal motion or highly impair it. As shown previously, the average velocity barely changes, while the error reduction has been greatly improved.

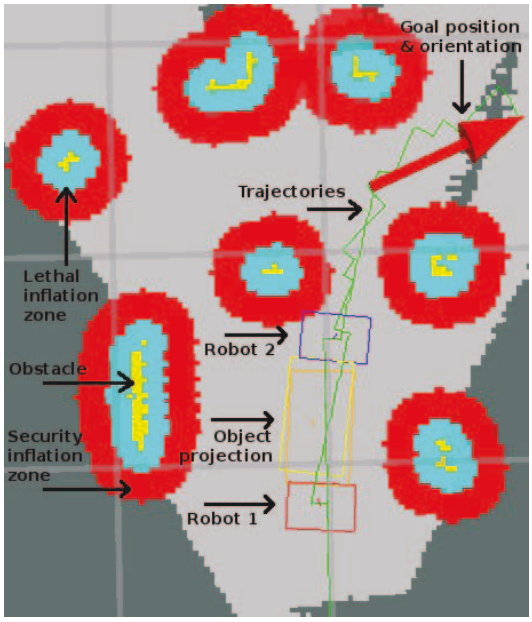


Fig. 7. Map of the starting and ending positions with obstacles, lethal and security inflation zones around them, the Nao and object footprints and goal position/orientation.

It is important to note that the ground in the testing room had sufficient texture and patterns to produce reliable data for the SLAM library. When tested on a plain ground, RTAB-Map could not, however, extract enough features for visual localization with the obstacles only.

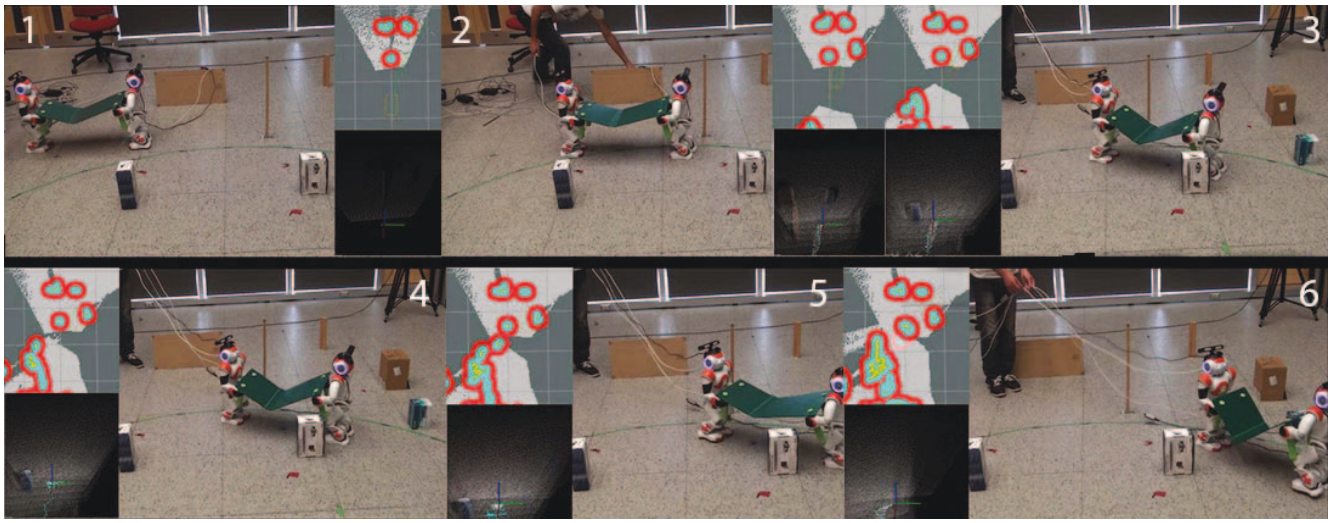


Fig. 8. Snapshots of the Naos navigating with an object

With the table linking the too robots together, significant drift cause by the visual odometry imperfection can make the Nao drift when navigating as shown in Fig. 8.

VII. Conclusion and Future Work

In this paper, we have proposed a system capable of carrying a long table with two humanoid robots while navigating in a cluttered environment; we also gave practical insights into the implementation of the proposed approach on a real humanoid robot. When moving throughout the environment, a depth camera and a SLAM library map the obstacles in real-time and provide a visual odometry. This information is then fused with each robot odometry to provide a consistent, continuous and reliable odometry data. Moreover, by controlling the hands adequately by using a whole-body control scheme, we were able to articulate the object in tight turns and to significantly reduce the lateral swing from propagating to the object and between robots.

In future works, in order to make the system more reactive and human-like, the arms should be used to absorb a part of the error instead of having to quickly move in fear of being unbalanced. In addition. the cameras in front of the Nao could be used to look at the other robot's relative position, instead of relying on hand position. As it is now, each robot can never really verify its relative position with respect to the other robot, and, as a result, it cannot detect drift errors in the visual odometry.

Acknowledgment

This research is supported by Natural Sciences and Engineering Research Council of Canada (NSERC), and partially by a collaborative research project funded by *"XIVe GROUPE DE TRAVAIL QUÉBEC-MEXIQUE"* and a Mitacs Globalink research internship.

References

[1] Yutaka Inoue, Takahiro Tohge, and Hitoshi Iba. Cooperative transportation system for humanoid robots using simulation-based learning. *Applied Soft Computing*, 7(1):115–125, 2007.

[2] Meng-Hung Wu, Atsushi Konno, and Masaru Uchiyama. Cooperative object transportation by multiple humanoid robots. In *System Integration (SII), 2011 IEEE/SICE International Symposium on*, pages 779–784, 2011.

[3] Kazuhiko Yokoyama, Hroyulu Handa, Takakatsu Isozumi, Yutaro Fukase, Kenji Kaneko, Fumio Kanehiro, Yoshihim Kawai, Fumiaki Tomita, and Hirohisa Hirukawa. Cooperative works by a human and a humanoid robot. In *Conference on Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International*, volume 3, pages 2985–2991, 2003.

[4] Antoine Bussy, Pierre Gergondet, Abderrahmane Kheddar, François Keith, and André Crosnier. Proactive behavior of a humanoid robot in a haptic transportation task with a human partner. In *RO-MAN, 2012 IEEE*, pages 962–967, 2012.

[5] Stephen G McGill and Daniel D Lee. Cooperative humanoid stretcher manipulation and locomotion. In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pages 429–433, 2011.

[6] Tomoaki Yoshikai, Takahiro Akimoto, Kaoru Kobayashi, Jumpei Tsuji, Hiroaki Yaguchi, and Masayuki Inaba. Achievement of 'mikoshi'with multiple humanoid robots as coordinated navigation problem based on real-time 3d space recognition in a dynamic environment. In *Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on*, pages 859–866, 2012.

[7] M. Likhachev, G. Gordon, and S. Thrun. ARA*: Anytime A* with provable bounds on sub-optimality. In *Advances in Neural Information Processing Systems*, volume 16, 2004.

[8] S. Oßwald, A. Hornung, and M. Bennewitz. Learning reliable and efficient navigation with a humanoid. In *IEEE Int. Conf. on Robotics and Automation*, pages 2375–2380, 2010.

[9] D. Maier, A. Hornung, and M. Bennewitz. Real-time navigation in 3D environments based on depth camera data. In *12th IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, pages 692–697, 2012.

[10] M. Labbe and F. Michaud. Online global loop closure detection for large-scale multi-session graph-based slam. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2661–2666, 2014.

[11] M. Labbe and F. Michaud. Rtab-map project on ros.org., 2014.

[12] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1620–1626, 2003.

[13] Antoine Rioux and Wael Suleiman. Humanoid navigation and heavy load transportation in a cluttered environment. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[14] D. Gouaillier, V. Hugel, P. Blazevic, C. Kilner, J. Monceaux, P. Lafourcade, B. Marnier, J. Serre, and B. Maisonnier. Mechatronic design of NAO humanoid. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 769–774, 2009.

LISTE DES RÉFÉRENCES

Aiyama, Y., Inaba, M. et Inoue, H. (1993). Pivoting : A new method of graspless manipulation of object by robot fingers. Dans *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. volume 1. p. 136–143.

Berger, E., Vogt, D., Haji-Ghassemi, N., Jung, B. et Amor, H. B. (2013). Inferring guidance information in cooperative human-robot tasks. Dans *IEEE-RAS Humanoid Robots (HUMANOIDS)*.

Bouguet, J.-Y. (2000). *Pyramidal Implementation of the Lucas Kanade Feature Tracker* (Rapport technique). Intel Corporation Microprocessor Research Labs.

Bussy, A., Gergondet, P., Kheddar, A., Keith, F. et Crosnier, A. (2012). Proactive behavior of a humanoid robot in a haptic transportation task with a human partner. Dans *Robot and Human Interactive Communication (RO-MAN)*. p. 962–967.

Defense Advanced Research Projects Agency (2016). DARPA Robotics Challenge., http ://www.theroboticschallenge.org/.

Dubins, L. E. (1957). On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, p. 497–516.

Escande, A., Mansard, N. et Wieber, P.-B. (2014). Hierarchical quadratic programming : Fast online humanoid-robot motion generation. *The International Journal of Robotics Research*, volume 33, numéro 7, p. pp. 1006–1028.

Ferreau, H., Kirches, C., Potschka, A., Bock, H. et Diehl, M. (2014). qpOASES : A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, volume 6.

Flash, T. et Hochner, B. (2005). Motor primitives in vertebrates and invertebrates. *Current Opinion in Neurobiology*, volume 15, numéro 6, p. 660–666.

Gouaillier, D., Hugel, V., Blazevic, P., Kilner, C., Monceaux, J., Lafourcade, P., Marnier, B., Serre, J. et Maisonnier, B. (2009). Mechatronic design of NAO humanoid. Dans *IEEE Int. Conf. on Robotics and Automation (ICRA)*. p. 769–774.

Harada, K., Hattori, S., Hirukawa, H., Morisawa, M., Kajita, S. et Yoshida, E. (2010). Two-stage time-parametrized gait planning for humanoid robots. *IEEE/ASME Trans. on Mechatronics*, volume 15, numéro 5, p. 694–703.

Harada, K., Kajita, S., Kanehiro, F., Fujiwara, K., Kaneko, K., Yokoi, K. et Hirukawa, H. (2007). Real-time planning of humanoid robot's gait for force-controlled manipulation. *IEEE/ASME Transactions on Mechatronics*, volume 12, numéro 1, p. 53–62.

Harada, K., Kajita, S., Saito, H., Morisawa, M., Kanehiro, F., Fujiwara, K., Kaneko, K. et Hirukawa, H. (2005). A humanoid robot carrying a heavy object. Dans *IEEE Int. Conf. on Robotics and Automation (ICRA)*. p. 1712–1717.

Hart, C. et Giszter, S. (2010). A neural basis for motor primitives in the spinal cord. *The Journal of Neuroscience*, volume 30, numéro 4, p. 1322–1336.

Hirai, K., Hirose, M., Haikawa, Y. et Takenaka, T. (1998). The development of Honda humanoid robot. Dans *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE. volume 2. p. 1321–1326.

Hirata, Y. et Kosuge, K. (2000). Distributed robot helpers handling a single object in cooperation with a human. Dans *IEEE Int. Conf. on Robotics and Automation (ICRA)*. volume 1. p. 458–463.

Hornung, A., Wurm, K., Bennewitz, M., Stachniss, C. et Burgard, W. (2013). OctoMap : An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, volume 34, numéro 3, p. 189–206.

Hornung, A., Wurm, K. M. et Bennewitz, M. (2010). Humanoid robot localization in complex indoor environments. Dans *IEEE/RSJ Intelligent Robots and Systems (IROS)*, IEEE. p. 1690–1695.

Inoue, Y., Tohge, T. et Iba, H. (2007). Cooperative transportation system for humanoid robots using simulation-based learning. *Applied Soft Computing*, volume 7, numéro 1, p. 115–125.

Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K. et Hirukawa, H. (2003). Biped walking pattern generation by using preview control of zero-moment point. Dans *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*. p. 1620–1626.

Kanehiro, F., Lamiraux, F., Kanoun, O., Yoshida, E. et Laumond, J.-P. (2008). A local collision avoidance method for non-strictly convex objects. Dans *2008 Robotics : Science and Systems Conference*.

Kanehiro, F., Morisawa, M., Suleiman, W., Kaneko, K. et Yoshida, E. (2010). Integrating geometric constraints into reactive leg motion generation. Dans *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. p. 4069–4076.

Kaneko, K., Kanehiro, F., Kajita, S., Hirukawa, H., Kawasaki, T., Hirata, M., Akachi, K. et Isozumi, T. (2004). Humanoid Robot HRP-2. Dans *IEEE International Conference on Robotics and Automation (ICRA)*. p. 1083–1090.

Kato, I. (1973). Development of WABOT 1. *Biomechanism*, volume 2, p. 173–214.

Kube, C. et Bonabeau, E. (2000). Cooperative transport by ants and robots. *Robotics and Autonomous Systems*, volume 30, p. 85–101.

Labbe, M. et Michaud, F. (2014a). Online global loop closure detection for large-scale multi-session graph-based SLAM. Dans *IEEE/RSJ International Conference onIntelligent Robots and Systems (IROS)*. p. 2661–2666.

Labbe, M. et Michaud, F. (2014b). RTAB-Map project on ROS.org., http ://wiki.ros.org/rtabmap.

Lawitzky, M., Mortl, A. et Hirche, S. (2010). Load sharing in human-robot cooperative manipulation. Dans *IEEE Robot and Human Interactive Communication(RO-MAN)*. p. 185–191.

Li, J., Huang, Q., Yu, Z., Chen, X., Zhang, W., Liu, H., Gao, J. et Duo, Y. (2015). Integral acceleration generation for slip avoidance in a planar humanoid robot. *IEEE/ASME Trans. on Mechatronics*, volume 20, numéro 6, p. 2924–2934.

Likhachev, M., Gordon, G. et Thrun, S. (2004). ARA* : Anytime A* with provable bounds on sub-optimality. Dans *Advances in Neural Information Processing Systems*. volume 16.

Maier, D., Hornung, A. et Bennewitz, M. (2012). Real-time navigation in 3D environments based on depth camera data. Dans *12th IEEE-RAS Int. Conf. on Humanoid Robots (HUMANOIDS)*. p. 692–697.

McGill, S. G. et Lee, D. D. (2011). Cooperative humanoid stretcher manipulation and locomotion. Dans *IEEE-RAS Int. Conf. on Humanoid Robots (HUMANOIDS)*. p. 429–433.

Miyata, N., Ota, J., Aiyama, Y., Sasaki, J. et Arai, T. (1997). Cooperative transport system with regrasping car-like mobile robots. Dans *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. volume 3. p. 1754–1761.

Nozawa, S., Maki, T., Kojima, M., Kanzaki, S., Okada, K. et Inaba, M. (2008). Wheelchair support by a humanoid through integrating environment recognition, whole-body control and human-interface behind the user. Dans *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. p. 1558–1563.

Ohmura, Y. et Kuniyoshi, Y. (2007). Humanoid robot which can lift a 30kg box by whole body contact and tactile feedback. Dans *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. p. 1136–1141.

Oßwald, S., Hornung, A. et Bennewitz, M. (2010). Learning reliable and efficient navigation with a humanoid. Dans *IEEE Int. Conf. on Robotics and Automation*. p. 2375–2380.

Ota, J., Miyata, N., Arai, T., Yoshida, E., Kurabatashi, D. et Sasaki, J. (1995). Transferring and regrasping a large object by cooperation of multiple mobile robots. Dans *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. volume 3. p. 543–548.

Rioux, A., Esteves, C., Hayet, J.-B. et Suleiman, W. (2015). Cooperative slam-based object transportation by two humanoid robots in a cluttered environment. Dans *IEEE Int. Conf. on Humanoid Robotics (HUMANOIDS)*.

Rioux, A. et Suleiman, W. (2015). Humanoid navigation and heavy load transportation in a cluttered environment. Dans *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*.

Rousseeuw, P. J. (1984). Least median of squares regression. *Journal of the American statistical association*, volume 79, numéro 388, p. 871–880.

Sakagami, Y., Watanabe, R., Aoyama, C., Matsunaga, S., Higaki, N. et Fujimura, K. (2002). The intelligent ASIMO : System overview and integration. Dans *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. volume 3. p. 2478–2483.

Scholz, J., Chitta, S., Marthi, B. et Likhachev, M. (2011). Cart pushing with a mobile manipulation system : Towards navigation with moveable objects. Dans *IEEE Int. Conf. on Robotics and Automation (ICRA)*. p. 6115–6120.

Shi, J. et Tomasi, C. (1994). Good features to track. Dans *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*. p. 593–600.

Shigemi, S., Kawaguchi, Y., Yoshiike, T., Kawabe, K. et Ogawa, N. (2006). Development of New ASIMO. *Honda R and D Technical Review*, volume 18, numéro 1.

Stasse, O., Davison, A. J., Sellaouti, R. et Yokoi, K. (2006). Real-time 3D SLAM for humanoid robot considering pattern generator information. Dans *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. p. 348–355.

Stilman, M. et Kuffner, J. (2004). Navigation among movable obstacles : Real-time reasoning in complex environments. Dans *IEEE Humanoid Robotics (HUMANOIDS)*. volume 1. p. 322 – 341.

Suda, R. et Kosuge, K. (2002). Handling of object by mobile robot helper in cooperation with a human using visual information and force information. Dans *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. volume 2. p. 1102–1107.

Takubo, T., Inoue, K. et Arai, T. (2005). Pushing an object considering the hand reflect forces by humanoid robot in dynamic walking. Dans *IEEE International Conference on Robotics and Automation (ICRA)*. p. 1706–1711.

Tomita, F., Yoshimi, T., Ueshiba, T., Kawai, Y., Sumi, Y., Matsushita, T., Ichimura, N., Sugimoto, K. et Ishiyama, Y. (1998). R&d of versatile 3d vision system vvv. Dans *International Conference on Systems, Man, and Cybernetics, 1998. IEEE*. volume 5. p. 4510–4516.

uQuadProg Solver (2009). http ://www.lis.deis.unical.it/~ furfaro/uquadprog/uquadprog.php.

Wang, Z., Admadabadi, M., Nakano, E. et Takahashi, T. (1999). A multiple robot system for cooperative object transportation with various requirements on task performing. Dans *IEEE Int. Conf. on Robotics and Automation (ICRA)*. p. 1226–1233.

Weisstein, E. W. (2005). Moore neighborhood. *From MathWorld–A Wolfram Web Resource. http ://mathworld. wolfram. com/MooreNeighborhood. html.*

Weisstein, E. W. (2012). von neumann neighborhood. from mathworld–a wolfram web resource.

Wu, M.-H., Konno, A., Ogawa, S. et Komizunai, S. (2014). Symmetry cooperative object transportation by multiple humanoid robots. Dans *IEEE Int. Conf. on Robotics and Automation (ICRA)*. p. 3446–3451.

Wu, M.-H., Konno, A. et Uchiyama, M. (2011). Cooperative object transportation by multiple humanoid robots. Dans *IEEE/SICE Int. Symposium on System Integration (SII)*. p. 779–784.

Yamashita, A., Arai, T., Ota, J. et Asama, H. (2003). Motion planning of multiple mobile robots for cooperative manipulation and transportation. *IEEE Trans. on Robotics and Automation*, volume 19, numéro 2, p. 223–237.

Yeon, J. S. et Park, J. H. (2014). A fast turning method for biped robots with foot slip during single-support phase. *IEEE/ASME Trans. on Mechatronics*, volume 19, numéro 6, p. 1847–1858.

Yokoyama, K., Handa, H., Isozumi, T., Fukase, Y., Kaneko, K., Kanehiro, F., Kawai, Y., Tomita, F. et Hirukawa, H. (2003). Cooperative works by a human and a humanoid robot. Dans *IEEE International Conference on Robotics and Automation (ICRA)*. volume 3. p. 2985–2991.

Yoshida, E., Blazevic, P., Hugel, V., Yokoi, K. et Harada, K. (2006). Pivoting a large object : whole-body manipulation by a humanoid robot. *Applied Bionics and Biomechanics*, volume 3, numéro 3, p. 227–235.

Yoshida, E., Esteves, C., Belousov, I., Laumond, J.-P., Sakaguchi, T. et Yokoi, K. (2008). Planning 3-d collision-free dynamic robotic motion through iterative reshaping. *IEEE Transactions on Robotics*, volume 24, numéro 5, p. 1186–1198.

Yoshida, E., Poirier, M., Laumond, J.-P., Kanoun, O., Lamiraux, F., Alami, R. et Yokoi, K. (2010). Pivoting based manipulation by a humanoid robot. *Autonomous Robots*, volume 28, numéro 1, p. 77–88.

Yoshikai, T., Akimoto, T., Kobayashi, K., Tsuji, J., Yaguchi, H. et Inaba, M. (2012). Achievement of 'mikoshi'with multiple humanoid robots as coordinated navigation problem based on real-time 3d space recognition in a dynamic environment. Dans *Humanoid Robots (HUMANOIDS), 2012 12th IEEE-RAS Int. Conf. on*. p. 859–866.

Zhang, Z. (1998). *A Flexible New Technique for Camera Calibration* (Rapport technique MSR-TR-98-71). Microsoft Research.